ISSN # 0748-9331

THE COMPUTER JOURNAL®

For Those Who Interface, Build, and Apply Micros

Issue Number 18

$2.50 US

Interfacing the Apple][

Parallel Interface for the Apple Game Port page 2

The Hacker's MAC page 8

S-100 Graphics Screen Dump page 10

The LS-100 Disk Simulator Kit page 13

BASE:

Part Six in a Series on

How to Design and Write Your Own Database page 18

Interfacing Tips and Troubles:

Communicating with Telephone Tone Control page 20

The Computer Corner page 22

THE COMPUTER JOURNAL
P.O. Box 1697
Kalispell, Montana 59903
406-257-9119

Editor/Publisher
Art Carlson

Art Director
Joan Thompson

Art Assistant
Lois Cawrse

Production Assistant
Judie Overbeek

Circulation
Donna Carlson

Contributing Editor
Ernie Brooner

Contributing Editor
Neil Bungard

Technical Editor
Lance Rose

The Computer Journal® is a monthly magazine for those who interface, build, and apply microcomputers.

The subscription rate for twelve issues is $24 in the U.S., $30 in Canada, and $48 airmail in other countries. Please make payment in U.S. funds.

Entire contents copyright © 1985 by The Computer Journal.

Advertising rates available upon request.

To indicate a change of address, please send your old label and new address.

Postmaster: Send address changes to: The Computer Journal, P.O. Box 1697, Kalispell, Montana, 59903-1697.

Address all editorial, advertising and subscription inquiries to: The Computer Journal, P.O. Box 1697, Kalispell, MT 59901

Editor's Page

It's Time to Return to The Bench

The microcomputer industry was started by techies who built and programmed their computers, but now most of the micros are purchased fully assembled and ready-to-go by business people. The industry has responded with appliance type micros designed for the usual business needs without requiring any technical knowledge from the operator. These systems may work well in their intended office use, but they do not fill the needs of the serious experimental computerist who wants to interface to strange devices which were not planned for during the original design. The older, less structured, systems with easily expanded I/O capabilites are actually more useful for controlling real world devices than most of the new systems.

One possible solution is to buy an older system with an open bus and build the boards to provide the additional functions. Another possible solution is to build peripherial devices which interface through the ports provided on the new systems. Either of these approaches requires someone to get their hands dirty on the hardware and to write the utility programs to make it work.

Hardware hacking (including the necessary programming) has been declining until recently, but it is growing again as new generations of computer users become interested in finding more uses for their micros. In answer to the many requests, we are planning more articles on designing, building, and programming computer add-ons, interfaces, dedicated controllers, and single board computers. We don't intend to abandon the areas that we have been covering, but we will increase the material on hardware construction, interfacing, and utility programs.

The microcomputer industry was started in garages where the initial ideas were converted into hardware, and the breakthrough software was written in the spare bedroom or on the kitchen table. Large organizations were formed to produce and market the products, but the conception came from individuals or small groups and not from structured organizations! In today's market it is difficult for the innovative individual to arrange sufficient financing to market a product, and the large organizations stifle creativity, that's why we are not seeing any new breakthroughs in the industry. The next significant advances will come from individuals with their hands on the hardware and on the keyboard, and not from corporate offices. Our goal is to provide the information and inspiration you need. It's up to you to get back to the bench and get your hands dirty.

System Specific or General?

One of the major problems in the design of interfaces and peripherials is the wide variety of computer port and bus structures. The lack of standardization requires that you either make your design specific for one particular model, which limits your market; or that you design it to work through a port common to many different computers, which can increase the expense and/or reduce its capabilities. This problem is very apparent when we select articles for *The Computer Journal*.

We want to present the information needed to solve the problems common to all systems, while demonstrating implementations with specific systems, and still provide useful information to people with a wide variety of systems. This is an almost impossible goal and one solution would be to change the magazine to cover only a specific microcomputer system, but I don't feel that a computer specific publication can provide the range of information available in a more general publication. Besides, I use two different systems now and would have a third system if I could find the time. I feel that com-

(continued on page 27)

INTERFACING THE APPLE][

Parallel Interface for the Apple Game Port

by Jan G. Eugenides and Tim Heffield

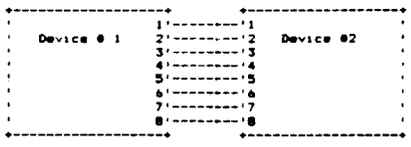
Ever wished you had just one more slot in your Apple? Wouldn't it be nice if you could interface printers, speech synthesizers, and other such devices without swapping cards? Well, now you can connect up to four parallel devices to the Apple game connector, all individually addressable, with this handy dandy little interface. And it'll cost you the princely sum of $30-$40 to do it!

This article is a collaboration between myself (Jan Eugenides) and Tim Heffield. Tim is the hardware expert, and he designed and built the interface card. I'm the software guy, and I wrote the printer driver (and this article!). The interface is very useful—I use mine to send data to my Epson printer, freeing up a slot for other uses.

The interface is made possible by the invention of a device called the UART, which stands for Universal Asynchronous Receiver Transmitter. The UART is a single chip which will convert serial data to parallel, and parallel to serial, all automatically. This makes it possible to design an interface for many types of equipment rather easily.

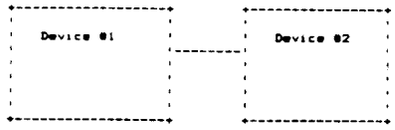
Parallel and Serial Data

There are basically two ways you can send a byte of information between two digital devices: parallel and serial. A *parallel* interface requires one line for each bit, therefore eight for one byte. Each device is connected to all eight lines, and an entire byte can be sent at one time. A parallel interface would look something like this:



This type of data transmission is very fast. It is used for short-run data transmission, such as between a computer and a printer, when it is feasible to run eight wires.

The other type of data transmission is *serial*. Each device is connected to only one line, and the information is sent one bit at a time. It has to be divided into separate bits at one end, and then reassembled into bytes at the other end. It requires that the two devices have some way of synchronizing to each other, and some way of recognizing the beginning and end of each byte. This is normally accomplished by the addition of one or more "control bits" to each byte that is sent. A serial interface would look something like this:



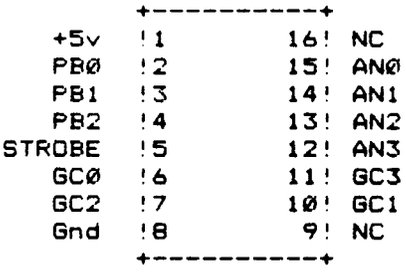
This type of data transmission is normally used for long-run data transmission, such as over telephone lines. Modems use serial transmission.

The Apple Game Connector

The Apple® computer does not have a built in parallel port, which is a bit inconvenient since most printers these days require parallel information. (At least the economical ones do!) However, the Apple does have a serial port, usually referred to as the game connector. The game connector actually consists of four one-bit outputs, called "annunciators," three one-bit inputs, and four analog inputs. Two of the one-bit inputs, and two of the analog inputs are used to connect the game paddles or a joystick to the computer. The other inputs and outputs usually remain unused, which is a shame because they provide the means for some very versatile interfacing. By using just one of the annunciator outputs and one of the one-bit inputs, it is easy to send serial data to a UART based interface, and the UART will convert it to parallel data which can be used by a regular Centronics-type printer, or any other parallel device!

Because there are four annunciator outputs and three one-bit inputs, it is actually possible to connect three UART interfaces to the game connector at once, with complete handshaking, and a fourth one without handshaking (I'll discuss the handshaking in detail a little later in this article). Furthermore, because the game connector also has a 5-volt pin, the interfaces can run off the Apple power supply.

Let's take time out for a minute to examine the game port a little more closely. Here's a diagram:



We are interested in AN0 through AN3 and PB0 through PB2. Here is a chart of the I/O locations for each of these:

	Location	Function
AN0	$C058	on
AN0	$C059	off
AN1	$C05A	on
AN1	$C05B	off
AN2	$C05C	on
AN2	$C05D	off
AN3	$C05E	on
AN3	$C05F	off
PB0	$C061	
PB1	$C062	*
PB3	$C063	

Note: if you have an older Apple reference manual, the chart on page 24 is wrong! These are the correct functions.

* Read these locations to determine the status. >$80 is on. <$80 is off

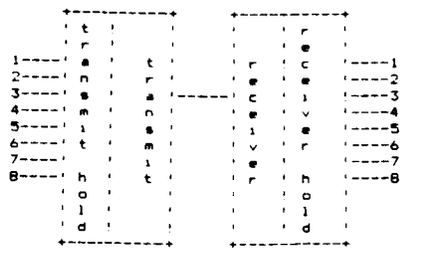
My printer driver only uses AN0 and PB0. As mentioned above, Tim designed the circuit board to allow more interfaces to be daisy-chained together, so by wiring them to AN1-3 and PB1-2 they can all be individually addressed. See the **"Construction"** section for more details.

Connecting the UART

The UART is a very simple device to use, and it is really not difficult to implement circuits based upon it. A

typical UART consists of two sections: a receiver and a transmitter, which are completely independent of each other except for the power supply and the control section. The transmitter section contains two one-byte registers, called the "transmitter hold register," and the "transmit register." When the DS line (data strobe) is brought low, the data on the eight input lines is loaded into the register. It is then transferred into the transmit register for subsequent assembling into bits for transmission. This is how you would convert parallel data to serial, but for our application we want to go the other way.

The receiver section is the reverse of the transmitter section. Serial data comes in and is assembled as a byte in the "receiver register." It is then transferred to "receiver hold register," which is connected to the eight data lines. These lines are tri-state so they can be connected directly to a data bus. Here is a diagram of a typical UART:



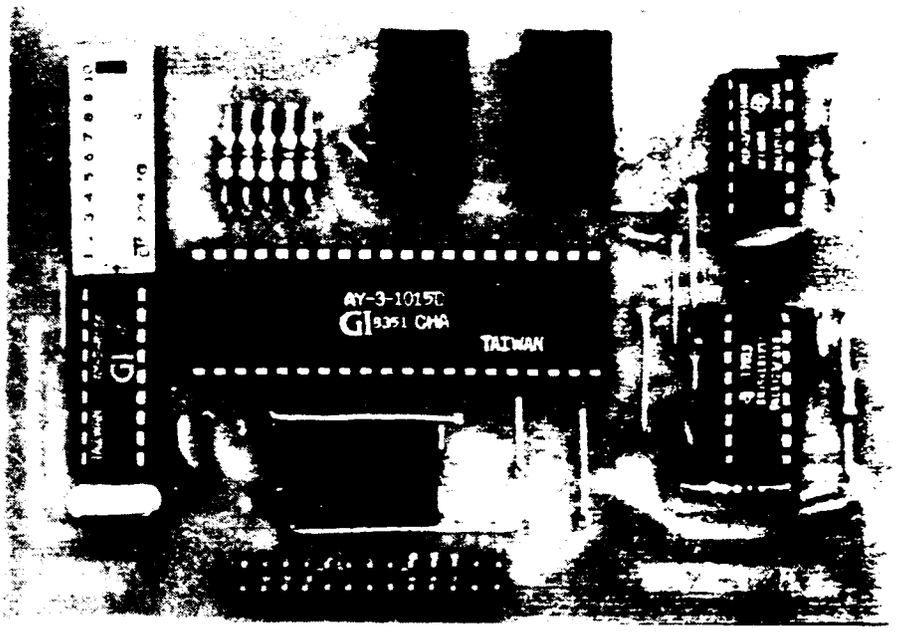
Transmitter section

```
DS..........Data Strobe
NB1, NB2.....Number of bits/Character
EPS.........Odd/Even parity select
TBMT........Transmit buffer empty
TSB.........Number of Stop bits
TCP.........Transmit clock
CS..........Control strobe
NP..........No Parity
SO..........Serial out
DB1-DB8.....Data bit inputs
```

Receiver section

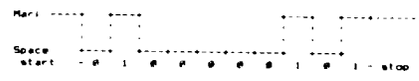
```
PE..........Parity error
FE..........Framing error
OR..........Overun error
RDE.........Receiver data enable
RDAV........Reset data available
RCP.........Receiver clock
SWE.........Status word enable
DAV.........Data available
SI..........Serial in
RD1-RD8.....Received data bits
//
```

The UART requires a clock at 16 × the baud rate, and certain programming commands. In Tim's circuit, another nifty little chip called a "baud rate generator" is used to provide the clock frequency. With the addition of some jumpers or a DIP switch, this makes it easy to set the interface for whatever baud rate you need. To use the printer driver included with this article, a rate of 2400 baud is needed. See Figure 1 for DIP switch settings.



Sending Serial Information

In order to send serial data to the UART through the game port, it is necessary to toggle the annunciator output at a specific rate, and in a specific pattern. To keep things comparatively simple, let's use a ten-bit pattern, with one start bit and one stop bit. The data line is normally held high until data is to be sent, then a zero bit (the start bit) is placed on the line for a specific time. This time depends upon the baud rate. For the 2400 baud rate we are using, it's about 417 microseconds. Then the eight bits of the character are sent, one at a time, beginning with the least significant bit. Finally, a one-bit (the stop bit) is sent, to signal the end of the character. For instance, the bit pattern for the letter "A" (ASCII $41) would be like this:



The hex value $41 equals %01000001 in binary (Apple uses the % symbol to indicate a binary number). After the start bit is sent (always a zero bit), then all the bits of the byte are sent in order as shown, and then the stop bit (always a 1) is sent. Obviously, the timing is fairly critical, and both devices must be running at very close to the same speed. There is a slight leeway for error, as long as the bits do not drift outside of the "frame."In order to assure the accuracy of a transmission, another bit, called the parity bit, is often added right before the stop bit. This bit is set in such a way that the overall number of one-bits in the

Baud Rate Selection

Baud	S1	S2	S3	S4
300	OFF	ON	OFF	ON
600	ON	OFF	OFF	ON
1200	OFF	OFF	OFF	ON
2400	ON	OFF	ON	OFF

UART Parameters

Parameter	Switch	Sw. Status
Odd Parity	S5	ON
Evn Parity	S5	OFF
7 Bits	S6	ON
	S7	OFF
8 Bits	S6	OFF
	S7	OFF
1 Stp. Bit	S8	ON
2 Stp. Bit	S8	OFF
Parity ON	S9	ON
Parity OFF	S9	OFF

Jumper Options

Jumper	Pt. to Pt.	Option
J1 Port Status	A > B	SW 0
	A > C	SW 1
	A > D	SW 2
	A > B	ANN 0
J3 Output Select	A > C	ANN 1
	A > D	ANN 2
	A > E	ANN 3
J2 +5V Select	Not installed if ext. +5V supply is desired instead of APPLE supply.	

Figure 1

character is either odd (called odd parity), or even (called even parity). In other words, for even parity in our example above, the parity bit would be zero, since the number of one-bits is already even. For odd parity, it would be set to one to make the number of one-bits odd. This provides a constant check on the validity of the data. If the parity is off, one or more bits in the byte are wrong. For a simple printer interface, parity is not necessary. Over noisy phone lines, however, it can be quite useful.

Circuit Description

A complete diagram is given in Figure 2. Operation of the circuit is fairly straightforward because most of the work is done by the UART, which is IC1. This is an AY-3-1015D UART, a very common and easy to obtain chip. See Figure 3 for a complete parts list, which includes the Radio Shack parts numbers for your convenience. The other main component is IC3 which is a crystal-controlled switch-selectable baud rate generator. Although Radio Shack has discontinued this item, all the stores we called in this area still had a half dozen of them in stock, so you shouldn't have much trouble getting one. This device is capable of producing sixteen different baud rates from 50 baud to 19,200 baud!! Figure 1 provides the switch settings for four of the most standard baud rates. The baud rate generator frequency (16 times the baud rate) is input to pin 17 of the UART. IC4 is a dual monostable multivibrator, which is used to widen the pulse created by the UART DAV (data available line), which is quite narrow, in order to assure that the computer can recognize that the printer is busy. This is the handshaking function I mentioned earlier, which is received by the Apple on the push-button inputs (PB0 through PB2). IC2 is used as a signal inverter in four different places in the circuit. It is interesting to note that this chip will accept a 12-volt swing, so you could conceivably add a 12-volt supply somewhere and create an RS-232 compatible interface. Food for thought, anyway.

Construction

Any standard method of construction can be used, such as perf-board or wire wrap, or a PC board can be made from Figure 4. Layout is not critical, other than keeping leads short for the crystal connections to IC3. Tim has designed a PC board for this project (see Figure 5 for the component layout). Unfortunately, we can only make one at a time in our limited home facilities, but we talked with the local PC board manufacturer, and if 70 to 80 of you want to donate $18.00 to the cause, we can have some run off professionally. There is a minimum run for this kind of thing, so we need at least 70 people to cover setup charges, etc. Please write us if you are interested, and if enough

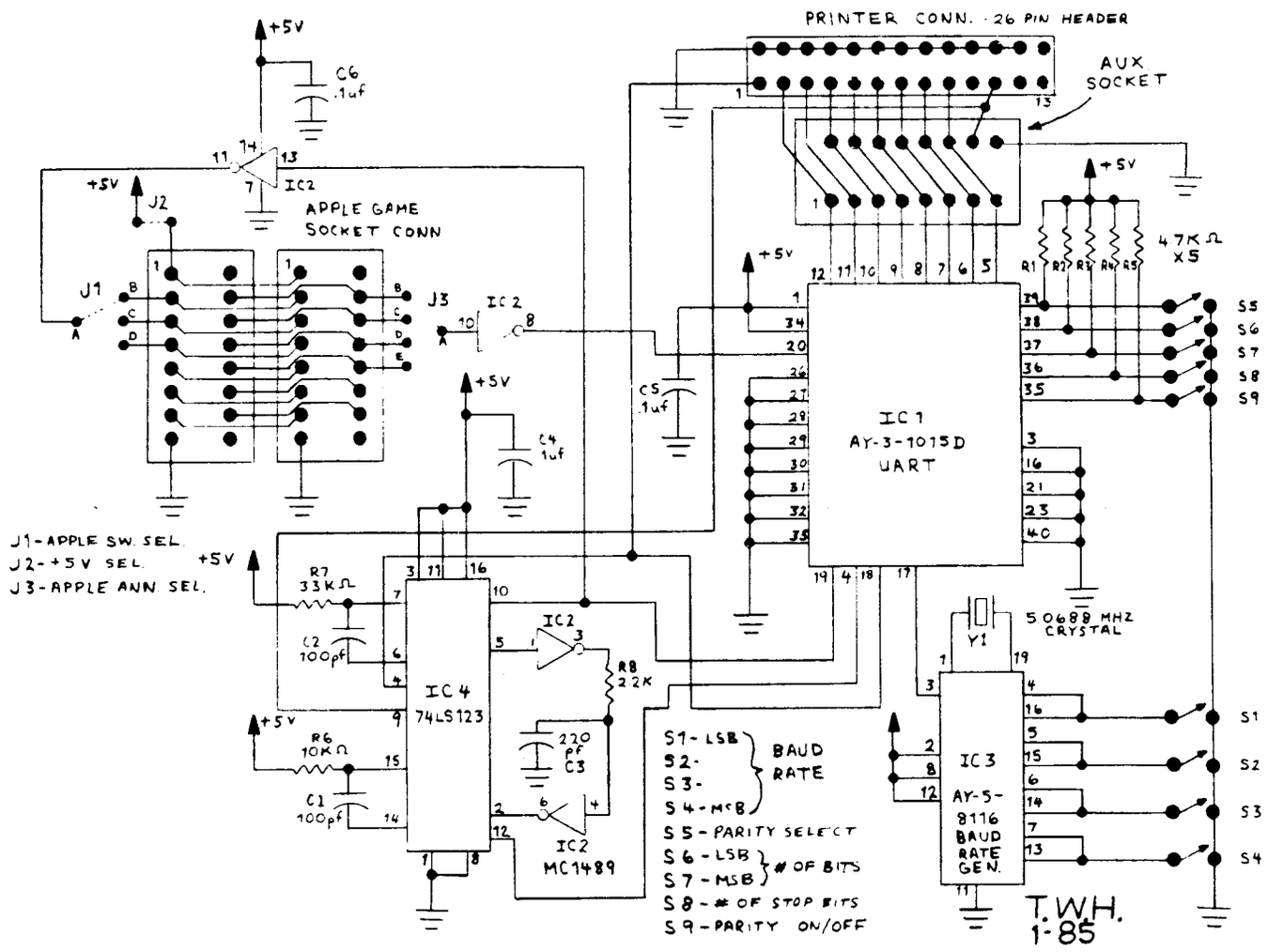


Figure 2

```
                Parts List

C1,C2 - 100 pf Capacitor
C3    - 220 pf Capacitor
C4-C6 - .1 uf Capacitor

R1-R5 - 4.7k 1/4 W Resistor
R6    - 10k  1/4 W Resistor
R7    - 33k  1/4 W Resistor
R8    - 2.2k 1/4 W Resistor

IC1   - AY-5-1015   (RS# 276-1794)
IC2   - 1489        (RS# 276-2521)
IC3   - AY-5-8116   (RS# 276-1795)
IC4   - 74LS123     (JAMECO)

Y1    - 5.0688 MHZ Crystal
        (JAMECO # CY-5.0688)
26 Pin Header       (JAMECO # 923863R)
16 Pin Dip Jumper Cable (RS# 276-1976)
(4ea.) 16 Pin IC Sockets
14 Pin IC Socket
18 Pin IC Socket
40 Pin IC Socket
9 Sw. or 10 Sw. - Dip Switch

Notes:
*(RS#)=Radio Shack Part Number
*IC1 and IC3 are not listed in 1985 RS
 catalog but many stores still stock them.
*A printer cable with 26 pin header plug
 and Centronics type connector is also
 needed. (RS# 26-1409) works well.
```

Figure 3

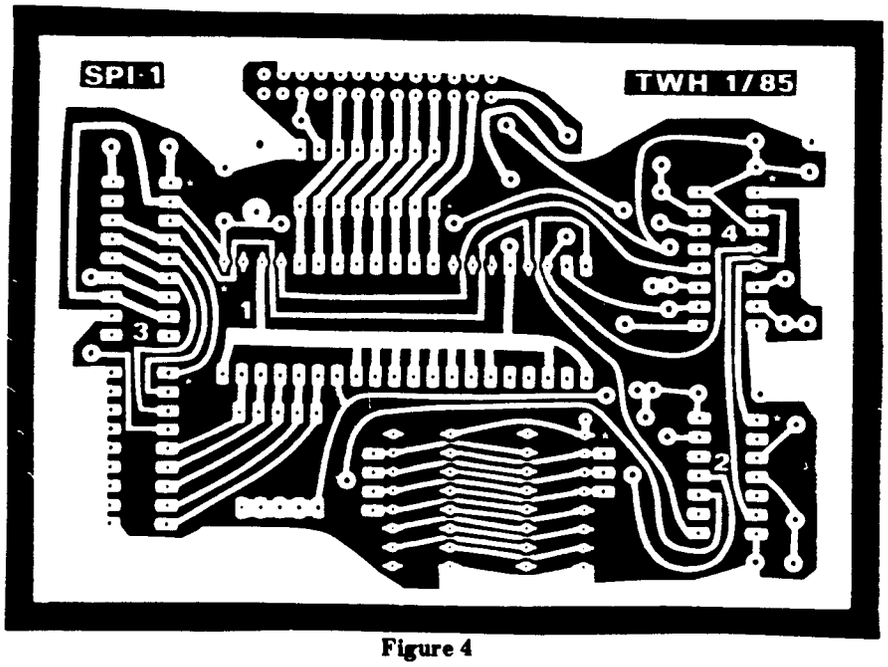


Figure 4

of us get together, we can do it.

Special Features of Our Board

The PC board Tim designed will fit nicely inside the Apple. Just use a couple of strips of double-sided foam tape to stick it to the right side, inside the case. Then run a short ribbon cable to the game connector. The board is designed with two 16-pin sockets in parallel, so you can still connect your game paddles or whatever. This is also how you would daisy chain two or three boards. Jumpers J1 and J3 allow you to configure the board for different game port inputs and outputs, so that you can retain individual control over each board.

The board receives its + 5 volts from the Apple game connector directly, but by removing jumper J2 you can use a separate supply if you prefer. The entire board only draws about 70 milliamps, so you don't have to worry about overtaxing the Apple supply, even if you connect more than one board.

Tim has used DIP switches to select the various baud rates and UART parameters, but you could hard-wire them if you know you won't need to change them, or if you want to save costs.

Tim has also included an auxiliary 16-pin socket which contains the 8-bits of parallel data and the handshaking line. This can be used to connect any other devices utilizing 8-bit parallel information, such as speech synthesizers, lamp dimmers, whatever. We'll be showing you how to interface the inexpensive Radio Shack speech synthesizer IC real soon, so watch this space!

The Printer Driver

Examine the code in Listing 1. This is the 6502 source code for a simple printer driver which will send data to the UART interface through the Apple game connector, which will then convert it to parallel and send it to the printer. Let's go through it line by line.

In order to redirect the Apple output from the screen, where it normally goes, to a printer or other device, the output hook at $36,37 must be changed to point to the desired device. When you type PR#1, for instance, the address of the card in slot one is placed there. PR#6 puts the address of the card in slot six there. We're not using a card, so we'll just put the address of our machine language program there, and all subsequent output will be directed to our routine, which can then pass it on to the UART through the game port.

Lines 1250-1310 perform this function. The JSR IOHOOK is a command to DOS to check the address in $36,37 and reset its own internal vectors to point to the address contained there.

FREE CATALOG AND SIGNAL PROCESSING BOOKLET

CP/M MSDOS

AFFORDABLE ENGINEERING SOFTWARE

TRSDC PCDO

CIRCUIT ANALYSIS
- Fast Machine Code
- Complete Circuit Editor
- Free Format Input
- Worst Case/Sensitivities
- Full Error Trapping

ACNAP Version 2.0 $69.95
- Any Size Circuit
- Input / Output Impedances
- Monte Carlo Analysis
- Transients (with SPP)

DCNAP $59.95
- Compatible Data Files
- Calculates Component Power
- 30 Nodes / 200 Components

SIGNAL PROCESSING

SPP $59.95
- Linear/Non-Linear Analysis
- FFT/Inverse FFT
- La Place Transforms
- Transient Analysis
- Time Domain Manipulation
- Spectra Manipulation
- Transfer Function Manipulation
- Editing and Error Trapping
- Free Format Input
- ASCII and Binary Files
- Fast Machine Code

VISA ● MASTERCARD

GRAPH PRINTING
- Linear/Logarithmic
- Multiple Plots
- Full Plot Labeling
- Auto/Forced Scaling
- Two Y-Axes
- ACNAP/SPP Compatible

PLOT PRO $49.95
- Any Printer
- Vertical/Horizontal

PC PLOT $59.95
- Screen Graphics
- Pixel Resolution
- Epson Printer

BV Engineering
Professional Software 2200 Business Way, Suite 207 • Riverside CA 92501 • USA (714) 781-0252

DOS always sits in between all the input and output in the Apple, sort of like a supervisor, and you'd better tell the supervisor what you're doing, or there will be trouble!

Once output begins arriving at our driver, several actions must be performed. In lines 1350-1370, the X, Y and A registers are saved so they can be restored later. Then, a check for a carriage return is made in line 1380. If one is found, it is sent, and then a linefeed ($8A) is sent. This is to make our driver perform the same as most parallel interface cards, which send linefeeds to the printer after a carriage return. Most printers can be set to insert their own linefeeds after a carriage return, so you could eliminate this step if you wish to configure your printer that way.

Line 1420 jumps to the output routine itself, and upon return from sending the character, the registers are restored in lines 1430-1450, and the routine exits through the Apple monitor routine COUT1 at $FDF0, which will print the character on the 40-column screen. You could replace line 1460 with a simple RTS if you do not desire output to the screen. While it is undoubtedly possible to route the data to the 80-column display, it is rather complex and decidedly slow, so I decided against it. You might have some fun trying to get it to work, though. (Let me know if you do!)

Line 1500 begins the actual output of the character through the game port. First, the character is saved on the stack, then the PRINTER READY line is checked. This is a line from the printer which is connected to the one-bit input at $C061. Normally low, this line will swing high when the printer is ready to accept the next character. Lines 1520-1540 check location $C061. As long as the value there is less than $80, the routine just waits. When the printer is ready, the character is retrieved in line 1550. This is the handshaking I told you about earlier.

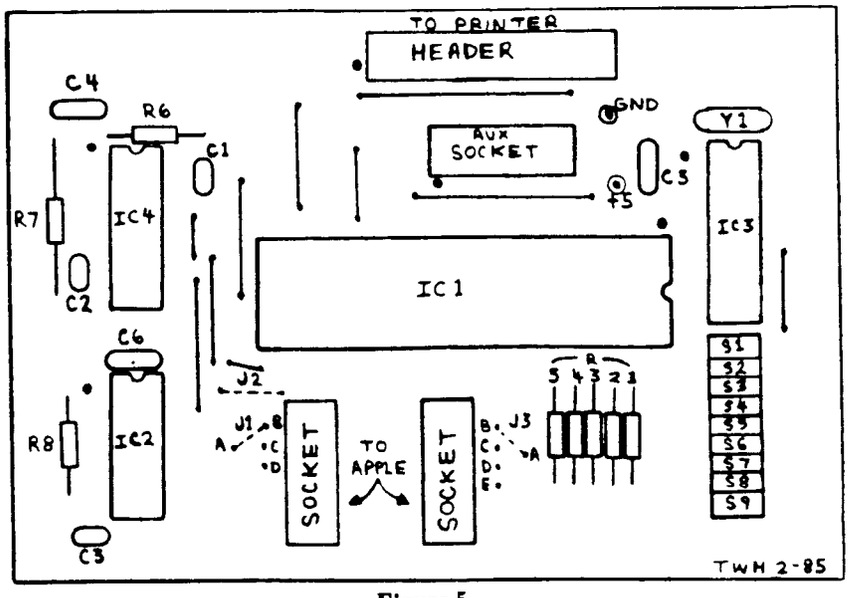


Figure 5

Listing 1

```
                        1000 *SAVE GAME.PORT.DRIVER
                        1010 *-------------------------------
                        1020 * 1/17/85
                        1030 *-------------------------------
                        1040 *Printer driver for game port interface
                        1050 *                by
                        1060 *          Jan G. Eugenides
                        1070 *         11601 N.W 18th St.
                        1080 *     Pembroke Pines, FL  33026
                        1090 *
                        1100 * Thanks to Bob Sander-Cederlof for
                        1110 * his help in Apple Assembly Lines
                        1120 *-------------------------------
03EA-                   1130 IOHOOK .EQ $03EA
03D0-                   1140 WMSTRT .EQ $03D0
FCA8-                   1150 WAIT   .EQ $FCA8
FDF0-                   1160 COUT1  .EQ $FDF0
                        1170 *
C058-                   1180 AN0.ON     .EQ $C058
C059-                   1190 AN0.OFF    .EQ $C059
C061-                   1200 READY.LINE .EQ $C061
                        1210 *-------------------------------
                        1220         .OR $300
                        1230 *       .TF PRINTER.DRIVER
                        1240 *-------------------------------
                        1250 CONNECT
0300- A9 0C             1260         LDA #DRIVER    Set output hook
0302- 85 36             1270         STA $36
0304- A9 03             1280         LDA /DRIVER
0306- 85 37             1290         STA $37
0308- 20 EA 03          1300         JSR IOHOOK
030B- 60                1310         RTS
                        1320 *-------------------------------
                        1330 *    PRINTER DRIVER
                        1340 *-------------------------------
030C- 8C 54 03          1350 DRIVER STY SAVEY       Save X and Y
030F- 8E 53 03          1360         STX SAVEX
0312- 48                1370         PHA            Save character
0313- C9 8D             1380         CMP #$8D
0315- D0 05             1390         BNE .1         Not CR, so no linefeed
0317- 20 29 03          1400         JSR POUT
031A- A9 8A             1410         LDA #$8A
031C- 20 29 03          1420 .1      JSR POUT
031F- AC 54 03          1430         LDY SAVEY      Restore X and Y
0322- AE 53 03          1440         LDX SAVEX
```

```
0334- 18        1580        CLC             Set up for start bit
0335- 48        1590 .1     PHA             Save char
0336- B0 05     1600        BCS .2          1-BIT, Send mark
0338- AD 59 C0  1610        LDA AN0.OFF       send zero bit
033B- 90 03     1620        BCC .3
033D- AD 58 C0  1630 .2     LDA AN0.ON        send mark (a one bit)
                1640 *--------------------------------
                1650 *   Timing loop for 2400 baud
                1660 *--------------------------------
0340- A9 09     1670 .3     LDA #$09    =2 Cycles
0342- 48        1680 .4     PHA         =3 cycles
0343- A9 20     1690        LDA #$20    =2 cycles
0345- 4A        1700 .5     LSR         =2 cycles
0346- 90 FD     1710        BCC .5      =3 cycles - loops 3 times = 14 cycles
                1720 *                  (BCC = 2 cycles if branch not taken)
0348- 68        1730        PLA         =4 cycles
0349- E9 01     1740        SBC #1      =2 cycles
034B- D0 F5     1750        BNE .4      =3 cycles
                1760 *--------------------------------
                1770 * 9 X 28 = 252 cycles
                1780 * add 4 cycles overhead
                1790 *  Total = 256 cycles = 250 microseconds
                1800 *
                1810 *
                1820 *--------------------------------
034D- 68        1830        PLA             Get char (Carry is set)
034E- 6A        1840        ROR             Next bit into carry
034F- 88        1850        DEY
0350- D0 E3     1860        BNE .1          Send next bit
0352- 60        1870        RTS
                1880 *--------------------------------
0353-           1890 SAVEX .BS 1
0354-           1900 SAVEY .BS 1

0000 ERRORS IN ASSEMBLY
```

Lines 1590-1630 do the actual sending of the bit. If the carry is clear, location $C059 is accessed. This turns off the annunciator output (we're using annunciator zero, or AN0 as it is called). If the carry is set, location $C058 is accessed. This turns on AN0. These locations are "softswitches," so the data read is not meaningful, only the accessing of the location is important. You could use BIT instead of LDA with the same result.

The Timing Loop

In lines 1670-1750 you'll see the timing loop for 2400 baud. In order to send 2400 bits per second, we need to send one bit every .417 milliseconds. In our driver, that means an extra delay of 250 microseconds is required. One machine cycle in the Apple takes 0.9799268644 microseconds, so we need 256 cycles to get 250 microseconds.

Finally, after the timing loop is done, line 1830 retrieves the character, and line 1840 puts the next bit in the carry. The Y register is decremented; if it is zero all bits have been sent, otherwise, the routine branches back to send another bit. Note that the carry is always set upon exiting the timing loop, this way the stop bit will always be a one, as required.

Conclusion

So, there you go—a parallel interface from the Apple game port. Let me know how you make out with it, and if you come up with any interesting ways of using it. Tim is busy working out a way to interface the Radio Shack speech synthesizer chip to it, so I'll probably write some kind of a driver for it and pass it on to you!■

Bibliography

"Build it Yourself," by Ralph Navarrete, *80Micro*, July 1983.

Apple Assembly Lines, by Bob Sander-Cederlof

"The UART," by Joseph J. Carr. *Computers and Programming*, Sept. 1981

FOR TRS-80 MODELS 1, 3 & 4
IBM PC, XT, AND COMPAQ

WHICH FORTH has all the POWERFUL APPLICATIONS?

- **DATAHANDLER** database
- **FORTHWRITE** word processor
- **FORTHCOM** communications
- **GENERAL LEDGER** accounting
- **GAMES** for fun and technique
- **EXPERT-2** expert system
- **TRADESHOW** commodities terminal
- **GRAPHICS, 8087** support, many other utilities

You've Been Thinking About It. Isn't It Time to Put It to Work?

MMSFORTH

The total software environment for IBM PC, TRS-80 Model 1, 3, 4 and close friends.

- Personal License (required):
  - MMSFORTH System Disk (IBM PC) $249.95
  - MMSFORTH System Disk (TRS-80 1, 3 or 4) 129.95
- Personal License (optional modules):
  - FORTHCOM communications module . . . . $ 39.95
  - UTILITIES 39.95
  - GAMES 39.95
  - EXPERT-2 expert system 69.95
  - DATAHANDLER 99.95
  - DATAHANDLER-PLUS (PC only, 128K req.) 99.95
  - FORTHWRITE word processor 175.00
- Corporate Site License Extensions . . . . . from $1,000
- Some recommended Forth books:
  - UNDERSTANDING FORTH (overview) . . . $ 2.95
  - STARTING FORTH (programming) 16.95
  - THINKING FORTH (technique) 16.95
  - BEGINNING FORTH (w MMSFORTH) . . . 16.95

Shipping/handling & tax extra. No returns on software.

Ask your dealer to show you the world of MMSFORTH, or request our free brochure.

MILLER MICROCOMPUTER SERVICES
61 Lake Shore Road, Natick, MA 01760
(617) 653-6136

THE HACKER'S MAC

by Lee Felsenstein

A View of the State of the Microcomputer Industry

We find ourselves in the midst of an industry which is struggling with the problems of adolescence. Many of us were instrumental in the exciting days surrounding the birth and infancy of the microcomputer industry, we have seen all the shakeouts, and we have to consider now whether we are still relevant to that industry.

Of course, this requires that I specify who is the "we" which I so glibly use. "We" are the people most strongly affected by what Steven Levy calls "the hands-on imperative" in his book, *Hackers*. We are underqualified, or overqualified, or just plain disrespectful of authority when it comes to things like the shape of the most important technology we know of. We insist on acting as if we had the right to try new ways of doing things, to have fun doing it, and to figure out later whether there's money in what we create.

Our biggest contribution to date has been the open microcomputer architecture. It's not a technical achievement, it's closer to a social achievement, and it should not be given up.

At this writing (January 1985), the microcomputer industry appears to resemble a dipole. At one end is the IBM architecture-open for now, and (they say), for the forseeable future. At the other pole is the Apple Macintosh. Anyone who wishes to compete in this market must identify themselves with one or the other.

When a species comes close to extinction and there are only a small number of survivors, that species is said to pass through a "genetic bottleneck," which will affect all the future generations if their numbers build up again. The diversity of the gene pool has been limited and with it the future adaptability of the species. The microcomputer industry is in danger of passing through such a bottleneck. At the other end we can forsee a limited number of options which reduce to:

"do it IBM's way,"
"do it Apple's way,"
or "go away."

Fortunately the iron laws of genetics do not apply to technology. New variations can be created with comparative ease, and we are the kinds of people who can carry out this creation. We know how to translate desire into phenomena.

An Interesting Metaphor

Imagine an old fashioned sci-fiction story describing the colonization of an uninhabited planet. The first settlers are renegades of various sorts, mostly evading various forms of authority. Under the pressure of necessity, they develop ways to work together without compromising their independence, resulting in some very interesting ways of doing things. As they gain a foothold on the planet, more arrive, only slightly less outrageous than their predecessors.

After a while the new arrivals include power brokers — people who live by telling other people what to do. They are looked upon with contempt by the settlers, but they bring their own servants with them, and they also bring various forms of capital, so they are left to do what they will. The power brokers go about setting up empires of various sorts.

Gradually these power brokers acquire the most developed parcels of land and establish agriculture of a sort which wouldn't work on the frontier and which requires the kinds of power structures which they deal in. There are fences put up and continuous territorial sniping goes on among the empires, punctuated by wars and skirmishes of various sorts. The empire — builders always, however, refer to the undeveloped areas of the planet as "wasteland" not fit for their attentions.

By this time many of the pioneer settlers are finding occasional or permanent employment with the power brokers, who come to control the market for the produce from the frontier and who need access to skills which the pioneers have developed. Manufacture is set up on a scale greater than subsistence, and many of the pioneers are employed as artisans and technicians as well as overseers, military scouts, etc. As a part of this process the pioneer settlers are forced out of their freeholdings in the developed area and many wind up in the new towns which are growing.

At a certain point the pioneers look around and realize as a group that they all work for the power brokers. Each step seemed reasonable, but the outcome does not. What do they do?

At this stage the story could take any of several courses. My preference is that the pioneers and some latecomers who are nonetheless good people take off one night and head for the wastelands, there to do it over again in the knowledge of what had happened once. Future chapters deal with the time when the two economies meet and come into conflict. Doubtless numerous books have been written around the same theme — it's a favorite of speculative fiction.

Using this metaphor to refer to the microcomputer industry, we have arrived at the time where the break-out is in order.

The Mac is Too Valuable to Be Left to Establishments

We are asked to believe that IBM has reformed its ways and will never again lock people out of their computers. We have no reason to believe these assurances. We also know that the user interfaces and performance of the IBM style computers are not growing better as the machines grow more complex. I am writing this on a CP/M Wordstar version and the cursor movements are effectively instantaneous in response to commands. My IBM Wordstar presents me with a serious lag in cursor movement, so that often I overshoot my

intended destination and must consciously slow down my usage. This, to me, is regression, and particularly galling since it is represented as technological progress, or at least an inevitable consequence of complexity.

To use Ted Nelson's phrase: Balderdash! Cybercrud!

The IBM architecture is based upon string-oriented I/O. The display is an imitation of a printer. It is slow, clunky, and deeply dissatisfying.

To this we contrast the Macintosh architecture. I typify it as having two-dimensional I/O: the mouse and the bit-mapped display image. This and its implications illustrate the major difference between the two architectures.

We need more innovation clustered around the Macintosh approach. We would certainly not entrust IBM with this innovation, and I see no reason why we should entrust any large (and growing larger) institution with it. That includes *Apple*.

War, it was said, is too important to leave to the military. The Mac, I say, is too important to leave to Apple. Nothing personal, Steve, it just follows from the nature of large organizations.

The Locus of Development

Since I believe that the Mac Architecture is worthy of our support, my reading of our experience says that we should create an open-architecture environment around the Mac. Such environments lead to a beehive (or wasps-nest) of software and peripheral development, which in turn defines what I call "the locus of development." This locus has inertia, and will tend to trap manufacturers into serving it. This is the lesson which IBM is still learning. Once your product depends on that cloud of bees, you had better not try to move it too far away from the hive.

I therefore propose that we develop a "slightly super" Mac, with an open architecture, but not too different from the Apple Mac. Ideally, programs written on this "Hacker's Mac" would be transportable to the Apple Mac with some loss of performance. This performance difference is important, since without it there is no tension exerted on the Apple design and our creature is meaningful only to us.

I further propose that the development process which we follow be as open as possible, with the tools in the public domain or available for low prices which cover overhead. This is the approach which we took in 1975-77 and it worked fabulously. It is this method of open development which I believe to be the real source of wealth for the microcomputer community.

Some Specifications

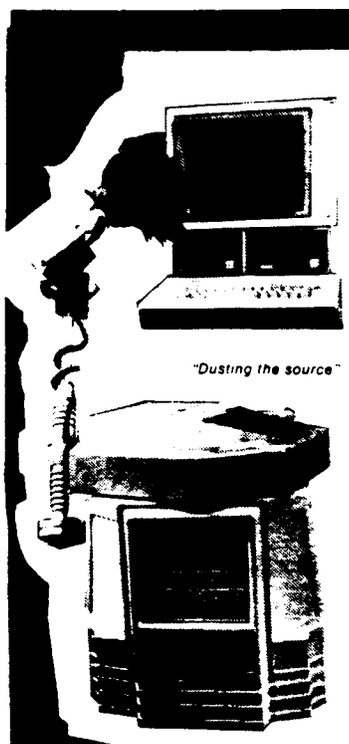
I will base my proposed specs on the following assumptions, in addition to the usual unspoken assumptions:

a. The minimum unit of individual computing is the terminal.

b. The best individual computers are highly expandable internally and can expand externally into multi-computer arrays and networks.

c. The best philosophy of configuration is that of providing just enough capability to encourage the users to learn their way into the use of that capability. This will tend to maximize the capability of the users.

(continued on page 23)


"Dusting the source"

HERO®/APPLE® HANDSHAKE

ROBI. . .an affordable interface for the robotics experimenter. Easy hook-up (8 screws on HERO ®, 1 card slot on Apple ® II or IIe) and a low price are combined with extra capabilities in the ROBI computer/ robot interface.

$199.00!

VISA MasterCard

ROBI SPECIFICATIONS
- *4 programmable bidirectional, 8-bit ports for interface and expansion*
- *programmable control over handshaking*
- *access to signals through tie point blocks on robot's Experimental Board*
- *6-foot cable for interface, limited remote operation*
- *user-friendly software quickly transfers files between computer and robot, stores and retrieves files to and from disk*
- *not copy protected. Software is provided in DOS 3.3*
- *liberally commented source code included*

BERSEARCH
Information Services

26160 Edelweiss Circle
Evergreen, CO 80439
(303) 674-0796

APPLE ® is a trademark of Apple Computer
HERO ® is a trademark of Heath Electronics

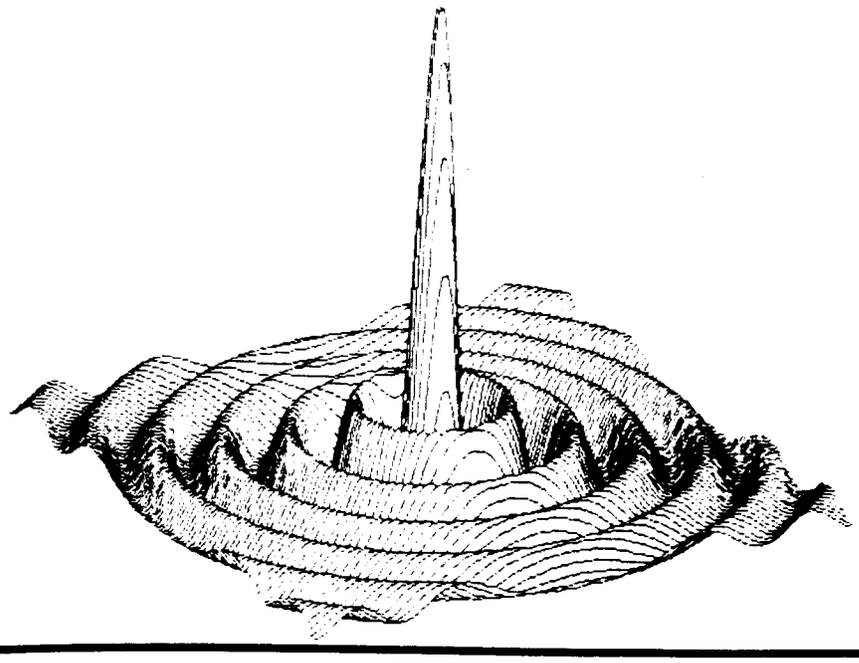
S-100 Graphics Board to Epson Screen Dump

by Lance Rose

For those hearty souls who built the high resolution graphics board described in Vol. II, Nos. 2,3, and 4 of *The Computer Journal* and who have an Epson or Epson-compatible printer, I'll show you how to take a graphics image on the video screen and dump it to the printer. This provides a very handy way to try out various combinations of image orientation and size without wasting paper and without the time-consuming printing process.

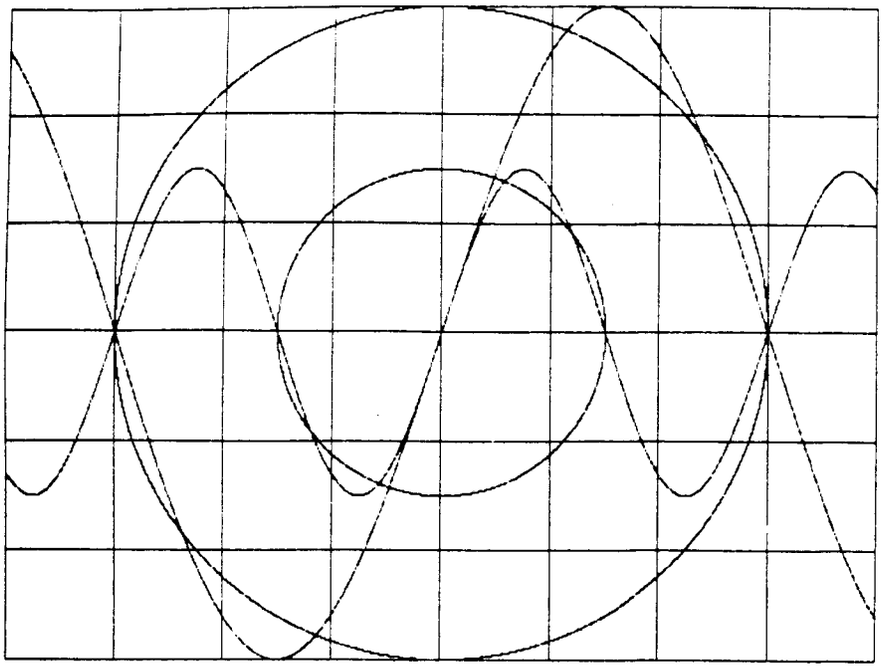
As a bit of a refresher, let me remind you that the graphics image is stored in 38400 bytes of RAM on the graphics board, with the RAM constantly scanned by supplementary circuitry to produce the video image. To transfer this to the printer, we need to identify a correspondence between a bit in the video RAM and a bit that will fire the appropriate printhead wire in the Epson. It turns out to be convenient to print the image rotated by 90 degrees from the way it is displayed on the screen. This comes about for two reasons. First of all, the 80-column versions of the Epson can only print 480 dots horizontally in normal graphics mode, while the graphics board displays 640 dots in that direction. Secondly, due to the nature of the printing process, the printhead wires are spaced 1/72 of an inch in the vertical direction while the horizontal spacing in graphics mode is 1/60 of an inch. This is due to the inherent difference between vertical spacing (6 or 8 line per inch) and horizontal spacing (10 or 12 characters per inch) in normal text. The increments of motion in each direction needed to produce these spacings just aren't very compatible between the two directions. Almost all printers are this way so don't feel too bad about it. For true x-y equality you really should have a plotter.

As a result of this, in order to produce a display on the printer that has both axes scaled the same, it is necessary to use only 400 dots in the vertical direction on the graphics board



Listing 1

```
TYPE      TXT
;         Program for dumping hi-res graphics board to Epson
;         CP/M-80 version, 1/24/85
;
VIDLO     EQU     100             ;Low byte of address port
VIDHI     EQU     VIDLO+1         ;High byte of address port
VIDIO     EQU     VIDLO+2         ;Data I/O port
;
          ORG     100H
;
          LXI     B,VSET          ;Vertical spacing string
          CALL    PRINT           ;Set Epson for 8/72" spacing
          LXI     H,79            ;Start at upper r.h. corner
LOOP:     CALL    LINE            ;Print line
          MVI     E,0AH
          CALL    WRTLST          ;Send linefeed here
          MVI     C,11
          CALL    BDOS            ;Look for key pressed
          ORA     A
          JZ      CONT            ;Keep going
          MVI     E,0FFH
          MVI     C,6
          CALL    BDOS            ;Input character without echo
          JMP     EXIT            ;Quit
CONT:     LXI     D,-1
          DAD     D               ;Back up for next line
          JC      LOOP            ;More to go
EXIT:     LXI     B,NSET
PRINT:    LDAX    B               ;Print message routine
          ORA     A
          RZ                      ;Zero terminates message
          MOV     E,A
          PUSH    B               ;Save string pointer
          CALL    WRTLST          ;Write to list device
          POP     B               ;Restore pointer
          INX     B               ;Go to next character
          JMP     PRINT
LINE:     LXI     B,GSET
          CALL    PRINT           ;Set Epson for graphics
LINE1:    MOV     A,H             ;Set up video RAM address
```



```
        OUT     VIDHI
        MOV     A,L
        OUT     VIDLO
        IN      VIDIO           ;Get the byte
        MOV     E,A
        CALL    WRTLST          ;Send to list device
        LXI     D,80
        DAD     D               ;Go to next video line
        MOV     A,H
        CPI     96H             ;Test for end of screen
        JC      LINE1
        SUI     96H
        MOV     H,A
        MVI     E,0DH
WRTLST: MVI     C,5             ;Write list function
BDOS:   PUSH    H               ;Save pointer
        CALL    0005H           ;Invoke BDOS
        POP     H               ;Restore pointer
        RET
;
VSET:   DB      0DH,1BH,'A',8,0 ;Return and set v.s. string
GSET:   DB      1BH,'K',224.1,0 ;Set graphics mode string
NSET:   DB      1BH,'2',0CH,0   ;Set normal and f.f. string
;
        END
```

Listing 2

```
VIDLO   EQU     100             ;Low byte of address port
VIDHI   EQU     VIDLO+1         ;High byte of address port
VIDIO   EQU     VIDLO+2         ;Data I/O port
;
        CSEG
;
        MOV     SI,OFFSET VSET  ;Vertical spacing string
        CALL    PRINT           ;Set Epson for 8/72" spacing
        MOV     BX,79           ;Start at upper r.h. corner
LOOP:   CALL    LINE            ;Print line
        MOV     DL,0AH
        CALL    WRTLST          ;Send linefeed here
        MOV     CL,11
        CALL    BDOS            ;Look for key pressed
        OR      AL,AL
        JZ      CONT            ;Keep going
        MOV     DL,0FFH
        MOV     CL,6
        CALL    BDOS            ;Input character without echo
        JMPS    EXIT            ;Quit
CONT:   SUB     BX,1            ;Back up for next line
        JNC     LOOP            ;More to go
```

(continued on page 15)

display. This will precompensate for the Epson's tendency to expand the display in the vertical direction (horizontal direction for the Epson since we are printing the image sideways). The logical procedure would be to use all 480 dots to try various images until you get what you like, then generate a screen with the vertical axis scaled down for printing.

Looking at the Epson manual, it turns out that in graphics mode each of 8 printhead wires is fired by a different bit in the byte being printed. Happily, this is the same scheme used in the video graphics board to turn on or off a single bit on the screen. By judiciously choosing which way to rotate the image for printing, we can use exactly the same byte to control the printhead wires that controls the electron beam in the CRT to turn a dot on or off. All the program has to do is read a vertical "strip" of bytes, 480 at a time, and send them to the printer. After 80 "strips" the image is completely printed.

Listing 1 shows the program necessary to do this with any computer running CP/M-80. It is a short assembly language program and should only take a few minutes to key in. If you're running an 8086 or 8088 under CP/M-86, Listing 2 shows a program that will perform the same function written in 8086 assembly language. Finally, for those who may have purchased or built a 68000 or 68008 board (see "Build a 68008 CPU Board for the S100 Bus" in Vol. II, No. 7 of *The Computer Journal*), Listing 3 gives a 68000 assembly language program listing for use with CP/M-68K. ■

FREE SOFTWARE

RENT FROM THE PUBLIC DOMAIN!

User Group Software isn't copyrighted, so there are no fees to pay! 1000's of CP/M and IBM software programs in .COM and source code to copy yourself! Games, business, utilities! All FREE!

CP/M USERS GROUP LIBRARY
Volumes 1-92, 46 disks rental—$45

SIG/M USERS GROUP LIBRARY
Volumes 1-90, 46 disks rental—$45
Volumes 91-176, 44 disks rental—$50
SPECIAL! Rent all SIG/M volumes for $90

K.U.G. (Charlottesville) 25 Volumes—$25

IBM PC-SIG (PC-DOS) LIBRARY
Volumes 1-200, 5¼" disks $200

174 FORMATS AVAILABLE! SPECIFY.

Public Domain User Group Catalog Disk $5 pp. (CP/M only) (payment in advance, please). Rental is for 7 days after receipt, 3 days grace to return. Use credit card, no disk deposit. Shipping, handling & insurance—$7.50 per library.
(619) 914-0925 information,(9-5)
(619) 727-1015 anytime order machine
Have your credit card ready! VISA, MasterCard, Am. Exp.

Public Domain Software Center
1533 Avohill Dr.
Vista, CA 92083

CTM

"...received my moneys worth with just one issue..."
—J. Trenbick

"...always stop to read CTM, even though most other magazines I receive (and write for) only get cursory examination..."
—Fred Blechman, K6UGT

U.S.A.	$15.00 for 1 year
Mexico, Canada	$25.00
Foreign	$35.00(land) - $55.00(air)
(U.S. funds only)	
Permanent (U.S. Subscription)	$100.00
Sample Copy	$3.50

CHET LAMBERT, W4WDR
1704 Sam Drive • Birmingham, AL 35235
(205) 854-0271

RP/M T.M.

By the author of Hayden's "CP/M Revealed."

New resident console processor RCP and new resident disk operating system RDOS replace CCP and BDOS without TPA size change.

User 0 files common to all users; user number visible in system prompt; file first extent size and user assignment displayed by DIR; cross-drive command file search; paged TYPE display with selectable page size. SUBMIT runs on any drive with multiple command files conditionally invoked by CALL. Automatic disk flaw processing isolates unuseable sectors. For high capacity disk systems RDOS can provide instantaneous directory access and delete redundant nondismountable disk logins. RPMPIP utility copies files, optionally prompts for confirmation during copy-all, compares files, archives large files to multiple floppy disks. RPMGEN and GETRPM self-install RP/M on any computer currently running CP/M®2.2. Source program assembly listings of RCP and RDOS appear in the RP/M user's manual.

RP/M manual with RPMGEN.COM and GETRPM.COM plus our RPMPIP.COM and other RP/M utilities on 8" SSSD $75. Shipping $5 ($10 nonUS). MC, VISA.

118 SW First St. - Box C
Warrenton, OR. 97146
Micro Methods, Inc.
(503) 861-1765

QUALITY SOFTWARE AT REASONABLE PRICES

CP/M Software by
Poor Person Software

Poor Person's Spooler **$49.95**
All the function of a hardware print buffer at a fraction of the cost. Keyboard control. Spools and prints simultaneously.

Poor Person's Spread Sheet **$29.95**
Flexible screen formats and BASIC-like language. Pre-programmed applications include Real Estate Evaluation.

Poor Person's Spelling Checker **$29.95**
Simple and fast! 33,000 word dictionary. Checks any CP/M text file.

aMAZEing Game **$29.95**
Arcade action for CP/M! Evade goblins and collect treasure.

Crossword Game **$39.95**
Teach spelling and build vocabulary. Fun and challenging.

Mailing Label Printer **$29.95**
Select and print labels in many formats.

Window System **$29.95**
Application control of independent virtual screens.

All products require 56k CP/M 2.2 and are available on 8" IBM and 5" Northstar formats, other 5" formats add $5 handling charge. California residents include sales tax.

Poor Person Software
3721 Starr King Circle
Palo Alto, CA 94306
tel 415-493-3735
CP/M is a registered trademark of Digital Research

THE LS-100 DISK SIMULATOR KIT

BY James O'Connor

It seems like only yesterday that upgrading a system from 4K of memory to 8K was a giant leap forward in processing power, not to mention a giant step backwards for our equipment budget. Perhaps more than any other item we purchase for our computer systems, the price of memory has declined steadily and sharply. The prime factors that have led to this are the emergence of dynamic memory technology as a reliable product, plus the fierce competition among American and Japanese manufacturers of dynamic memory ICs.

Dynamic memory, reduced to its simplest explanation, is a collection of capacitors which are charged or not charged so as to represent data. The problem with capacitors is that as soon as they are charged up they immediately start to discharge and, if nothing were done about this, they would very quickly forget everything stored in them. To solve this problem, dynamic memory circuits provide a refresh signal which serves to recharge the capacitors without altering what is stored in them. In the early days this refresh signal was provided by separate logic circuits but today many microprocessors (such as the Z-80) inherently provide this signal. Another early problem for dynamic memories was excessive susceptibility to noise, much of which was generated by all those charges being stored and read out. Proper printed circuit board design was essential for correct operation. Today these design principles are well understood and almost all modern dynamic memory boards are just as reliable as static memory systems. A major plus for dynamic memories is that they have always occupied less space, consumed less power and thus produced less heat than static memories.

It wasn't long before it occurred to some clever people that dynamic memory was now practically and economically capable of taking the place of, or at least complementing, some of the basic peripherals of a computer system such as the disk drives. Now why would you want to simulate a disk drive with memory chips? Because memory chips are electronic devices, they operate at electronic speed — we all learned in Elementary Physics that electrons travel at the speed of light. Disk drives, on the other hand, are electro-mechanical and from Elementary Logic we learned that the top speed of any device is the speed of its slowest part (the mechanical aspect of disk drives.) The other reason is that almost all software, both operating systems and applications programs, understand tracks and sectors as opposed to memory locations for storing large amounts of data. Thus a memory that simulates a disk drive should work with almost all current software, while providing a most impressive boost in speed.

Disk simulators are an unquestioned success but most of the early ones cost over a thousand dollars for about 256K of memory, which is roughly equivalent to the capacity of a single sided, single density 8 inch diskette (240K bytes). For those of us on a limited budget, and who only use our systems occasionally, that can be too costly to justify a disk simulator.

Thus I was intrigued when I noticed an ad from Digital Research Computers (DRC) of Garland, Texas 75046 (not to be confused with the software company of Pacific Grove, California) for the Light Speed 100 (LS-100). This product is available either as a 256K kit for $229.00 or as a bare board with software for $69.95. I quickly scraped together the necessary funds and ordered the full kit, which arrived about 6 weeks later. I suspect that demand has been brisk since this kit represents a substantial value to price ratio.

The Kit

Upon opening the package I found that the parts were all present but were not separated into little envelopes as is the case with some other kit makers. You must be able to correctly identify electronic parts with this kit. There is a neatly typed assembly/usage manual of nineteen pages. DRC has wisely produced the schematic on an 11 × 17 inch page so that it is fully legible. It is also surprisingly simple because of the use of large scale ICs in the design of this board. A single sided, single density 8″ diskette containing the software completes the package. I don't know if any other disk size or format is available, so you should contact DRC first if you cannot read this format or find someone who can transfer it to your format.

The circuit board for this kit has double sided traces with plated-through holes and a solder mask to avoid solder bridges between the closely spaced traces. The board appears to have been laid out by a computerized drawing system as the traces are very neat and machine-like in design. Part locations are outlined and identified in white on the top of the board. The parts in the kit were all of good quality. This board uses 64K × 1 bit dynamic memories (4164s), 32 of them, and these were very tightly piggybacked on top of each other in tiny little packages and then wrapped in aluminum foil to protect them from static damage. There are also 51 tiny little ceramic bypass capacitors and 39 16-pin DIP sockets — as with any memory type board there are a lot of parts.

The assembly instructions occupy three pages of step by step procedures. There is a parts layout diagram in the back of the manual, but no other drawings or pictures of the completed card are included.

Construction

Whenever I assemble a kit I find that they fall into one of two categories. The first type seems to struggle against its own construction — nothing goes easily or smoothly, but with perseverance it

eventually gets built. With the second category it almost seems that if you could shake the box long enough, everything would just naturally fall together. The LS-100 fits into the "shake the box" category; even though there are a lot of pins to be soldered it all went smoothly, and inexorably resulted in the finished board. It took me about two evenings to complete, but I was rushing it in anticipation of the improvement a disk simulator would make in my system.

Setup consists of setting two 8 position DIP switches, one to address the board (which uses four consecutive I/O ports), and the second to select the board. You can install up to eight of these boards in a system (2 MegaBytes!!). Each additional board would use the same four I/O port addresses, so the second DIP switch numbers each board to permit individual selection.

There are three jumpers to set. One jumper selects whether or not you plan to provide standby power to the circuit when your computer is off. The next jumper selects Advanced or Normal Ready; there is a half page explanation of this in the manual which didn't really explain it to me. I would have had to research this somewhere else except that the instructions say to select the Advanced option and run the diagnostic program — if there are no errors, then your system can handle that option. Otherwise you must use the Normal Ready. The last set of jumpers allows you to indicate whether your processor provides a refresh signal (Z-80) or if it doesn't (8080), and what bus line carries the signal at what polarity. This explanation requires you to know something about your own system. The documentation contains a transposition error in this section. If the refresh signal occurs on bus pin 65 you should jumper J9, not J8 as instructed, and if refresh is on pin 66, jumper J8 not J9. There is one sample page of jumper settings for an 8080 system and one for a Z-80 system, and on these, the jumper options appear to be correctly labeled (further proof of the value of examples in documentation).

Once I had chosen the options for my system I installed the board, ran the diagnostics, and it worked right away. I haven't had to touch it since.

Software

There are three separate software programs for this board; a diagnostic program, a format program, and an install program. They are all supplied on the disk in BOTH source and object form. They are designed for use with CP/M-80. The full documentation is also supplied on the disk so you can print out extra copies of the manual if you wish (don't neglect to fix the transposition error I mentioned). This is a really neat feature that I wish all manufacturers would routinely adopt whenever possible. I often lose or misplace manuals but I can almost always find a disk.

The diagnostic program does exactly what its name implies and is used once to verify the operation of the board and the jumper option for Advanced or Normal Ready, then afterwards only if you suspect some problem with the board. The format option is designed to fill the memory on the board with a data pattern exactly like a freshly formatted disk. Each time you turn on power you must run the format program, as the memory will contain random data. You can avoid this by providing stand-by power to the board, but it takes less than a second to format a board, thanks to electronic speed. Both of these programs can be run as is, provided you use the default port addresses. If you need other addresses then you must change the equates in each program and reassemble with CP/M's ASM.COM or equivalent.

The third program is the driver, which serves to translate the sector and track addresses of a disk drive to memory locations within the LS-100. It also provides a checksum feature to verify the accuracy of the stored data. When the driver is installed your system gains a new disk drive. The default is drive "E:" which worked fine on my system, but you can change this if you already have a drive E:.

It is unlikely that you can use the pre-assembled version of the installer/driver, as it must be tailored to your system size. It is really two separate programs which are merged together into one COM file; one part installs the other part (the LS-100 driver) into your operating system. You may need to edit these files so as to properly set the memory size of your operating system as well as the I/O addresses if you can't use the defaults. Then you use DDT or ZSID to merge the two programs together. The reason for this is that the driver is assembled to operate at an address just below your operating system in high memory. It is the job of the install portion to move it there and then modify the operating system jump table to reference it. If you tried to load the driver at its true address it would result in a very large COM file indeed. By combining it with the installer you get a much more compact program, plus it can't just load, it must alter some addresses in the operating system itself.

This sort of procedure will be old hat to anyone who has ever modified their CBIOS. I have done this procedure many times but I found that even after reading the explanation of it in the LS-100 manual, I didn't realize that's what it was instructing me to do. The description of this process is somehow disjointed and disorganized. It mixes the process of creating the combined install/driver with an explanation of how they work. In essence, it is all explained but could use more clarity. There is a console listing example of the procedure, but be warned — you cannot copy it line for line if your system is a different total memory size — you must figure out where and how to alter it.

An alternative is to put the driver code directly into your CBIOS; the DRC manual describes the general theory of how to do this. This alternative would require that you know and understand your system's CBIOS. It is not possible for DRC to do more than provide a theoretical explanation (albeit a quite good one, as they use examples of the code needed to do this).

There is one problem that the manual makes no mention of and that could be disastrous on certain systems. The DRC software uses one byte at address 40 hex as a signal to indicate that the driver program has been installed. The locations from 40 to 4F hex are designated by CP/M as being available to the CBIOS for storing variables or switches as needed, and many do so. Typically some of these bytes will describe the nature and type of disks installed and status of certain operations. If your CBIOS already uses

the byte at 40 hex, you will have two different parts of the operating system using the SAME location for different purposes, each thinking it is the only user of that byte. This could cause big trouble if the byte contains disk status, as your operating system might clobber a disk because of the data stored there by the LS-100. On my system it isn't as serious, as it is the operating system that clobbers the LS-100 byte value, so that every so often the LS-100 "disappears" (as far as the operating system is concerned) and I have to run the install program to make it reappear.

To solve this problem you can do as follows: If you have the source code for your CBIOS then check to see if it uses this location. Be careful though, because if it stores a word value (two bytes) at location 41 hex then it *does* use location 40 while not explicitly addressing that location. If you find this to be the case then you can either change your operating system (not recommended, as you would have to change every copy of it), or just change the location the LS-100 uses to one that your operating system doesn't use, between 40 and 4F hex. If you don't have the source code then you will have to contact the vendor who supplied your CBIOS. If that's not possible then you may want to change the LS-100 to use location 4F and run a lot of tests to insure that it works OK. What have I done? Well, nothing yet because my CBIOS uses ALL of the available locations and besides, I plan to integrate the driver into my CBIOS, eliminating the need for that byte (if the driver is always present you don't need a byte to tell it that).

One nice feature of the software is that if you add more LS-100 boards to your system, the programs will automatically detect this and adjust accordingly. However, if you have integrated the driver into your CBIOS then you will have to change the Disk Parameter Block to take advantage of the additional boards.

Using the LS-100

If you haven't integrated the driver into your operating system, you must first format the LS-100 and then install the driver each time you power up. I have made up a little SUBMIT file to do this. If you know how to do it, you can

(Graphic listing continued from page 11)

```
EXIT:   MOV     SI,OFFSET NSET
        CALL    PRINT
        RETF
PRINT:  MOV     DL,[SI]          ;Print message routine
        OR      DL,DL
        JNZ     PRINT1
        RET                      ;Zero terminates message
PRINT1: PUSH    SI               ;Save string pointer
        CALL    WRTLST           ;Write to list device
        POP     SI               ;Restore pointer
        INC     SI
        JMPS    PRINT            ;Go to next character
LINE:   MOV     SI,OFFSET GSET
        CALL    PRINT            ;Set Epson for graphics
LINE1:  MOV     AL,BH            ;Set up video RAM address
        OUT     VIDHI,AL
        MOV     AL,BL
        OUT     VIDLO,AL
        IN      AL,VIDIO         ;Get the byte
        MOV     DL,AL
        CALL    WRTLST           ;Send to list device
        ADD     BX,80            ;Go to next video line
        CMP     BH,96H           ;Test for end of screen
        JB      LINE1
        SUB     BH,96H
        MOV     DL,0DH
WRTLST: MOV     CL,5             ;Write list function
BDOS:   PUSH    BX               ;Save pointer
        INT     224              ;Invoke BDOS
        POP     BX               ;Restore pointer
        RET
;
        DSEG
;
        ORG     100H
;
VSET    DB      0DH,1BH,'A',8,0  ;Return and set v.s. string
GSET    DB      1BH,'K',224,1,0  ;Set graphics mode string
NSET    DB      1BH,'2',0CH,0    ;Set normal and f.f. string
```

Listing 3

```
*
VIDLO   EQU     $FFFF00+100      *Low byte of address port
VIDHI   EQU     VIDLO+1          *High byte of address port
VIDIO   EQU     VIDLO+2          *Data I/O port
*
        TEXT
*
        MOVE.L  #VSET,A0         *Vertical spacing string
        BSR     PRINT            *Set Epson for 8/72" spacing
        MOVE.W  #79,D2           *Start at upper r.h. corner
LOOP:   BSR     LINE             *Print line
        MOVEQ   #$0A,D1
        BSR     WRTLST           *Send linefeed here
        MOVEQ   #11,D0
        BSR     BDOS             *Look for key pressed
        TST.W   D0
        BEQ     CONT             *Keep going
        MOVEQ   #$FF,D1
        MOVEQ   #6,D0
        BSR     BDOS             *Input character without echo
        BRA     EXIT             *Quit
CONT:   SUBQ.W  #1,D2            *Back up for next line
        BCC     LOOP             *More to go
EXIT:   MOVE.L  #NSET,A0
PRINT:  MOVE.B  (A0)+,D1         *Print message routine
        BNE     PRINT1
        RTS                      *Zero terminates message
PRINT1: MOVE.L  A0,-(SP)         *Save string pointer
        BSR     WRTLST           *Write to list device
        MOVE.L  (SP)+,A0         *Restore pointer
        BRA     PRINT            *Go to next character
LINE:   MOVE.L  #GSET,A0
        BSR     PRINT            *Set Epson for graphics
LINE1:  ROL.W   #8,D2            ;Set up video RAM address
        MOVE.B  D2,VIDHI
```

(continued on page 27)

also cause this SUBMIT to execute automatically upon startup (this requires a small patch to the CP/M Operating System.) It takes the format and install programs combined less than two seconds to accomplish their tasks.

In operation, the LS-100 definitely improves the performance of any program that uses overlays, plus it is much quieter. Compilations and long assemblies are much faster. But there is a negative side—you must transfer files to the disk simulator prior to working on them. I have found that this step can take some time and effort as it frequently involves more than one or two files. Even more disconcerting is the fact that you can't always use wild-card names to process them all (the Public Domain SWEEP programs are really useful here). Time spent here can cancel out the time saved if you'll only be working for a brief period. The second, and perhaps more sinister problem, is that you must remember to transfer the files BACK to diskettes—otherwise all that work evaporates when you turn off the power. I have already had the experience of shutting down the system and, just as the little dot on the CRT was fading away, realizing that I hadn't transferred the files. Argghhh and double argghh!!! I have only done that once, so perhaps one must do it at least once in order to avoid it. These problems are generic to all disk simulators.

I think this board should really have stand-by power to realize its full potential, but that is easier said than done. The board draws about 600 milliamps which is a lot to expect from a battery. DRC recommends a filtered DC power supply of between 8 and 10 volts at 800 milliamps because the stand-by power goes through the on-board regulator and must be at least 8 volts for correct operation. A suitable supply could be made by following numerous examples of such circuits or by buying something similiar. However, such a unit will draw enough power that you should consider the impact on your electric bill, especially if you won't use your system for periods of several days. It would be cheaper to back up the files onto diskettes and then turn off the power, which you should do in any case for protection from power failures. The ideal would be a supply that feeds power from a battery that is constantly trickle charged from the AC line; this supply could ignore brief power interruptions. Perhaps some reader could design a suitable circuit for a future issue—it would be useful for many more applications.

Who Should Buy This Kit?

First, you should have an S-100 bus system with at least one available slot. You will need to know how to solder, and must be able to identify electronic parts. If you are a novice builder and know someone who can help you with this, you should be able to assemble this kit if you take your time; there are no tricky or difficult assembly steps.

You must also be able to adapt the software to your system. This requires that you know or be able to find out what port addresses are already used in your system and where to locate the signal byte mentioned earlier in this review. You will need to be able to edit the source files and reassemble them, and use DDT to combine the install/driver programs. None of this is very complex, unless you've never done it before. Again, try to find someone to guide you if necessary. One reason I like to build kits is that they force me to extend my knowledge into areas I would otherwise never enter. This kit is just right for anyone who has yet to modify some software; it is not too difficult, there are lots of people (especially in User Groups) who can help, and the results can be most worthwhile.

Should you want to integrate the driver into your CBIOS then you will need all the above skills plus the SOURCE code for your CBIOS. This may be a problem, as not all companies supply the code even if you ask them for it. And if you've never fiddled with a CBIOS you will need skilled guidance or a good book or two to help with the process. Don't rule out the LS-100 if you can't do this, as this step is only a matter of convenience, and it works just fine without it.

The bare board is available for those who want to obtain their own parts. One IC, the Intel 8203-1 is rare and costly so DRC also sells it. Obviously, bare board building is for the more experienced kit basher, but if you want to try it this is a good unit to start with. The board costs $69.95 and includes the software disk. The 8203 IC is $29.95.

Summary

This is a good kit; it is well designed and includes quality parts. The documentation could use some improvement, but it is certainly adequate in that the needed information is there, even if you must sometimes study the text. I didn't need to contact DRC but I understand that they have been in business for some time, which usually indicates that a firm supports their products.

The resulting disk simulator can really improve the performance of a system (I am using it to prepare this review). And if you want more than 256K of electronic disk space you can easily add additional LS-100's (up to eight depending on the number of available slots in your system). Since I purchased the kit for $399, DRC has reduced the kit price to $229, probably due to reduced cost of memory ICs. Perhaps, too, the six week delivery has improved—I ordered just after the kit was advertised, so it may have taken DRC time to ramp up production.

Most of the problems you're likely to encounter with this product are really caused by the wide variance between S-100 bus CP/M systems, and the fact that any disk simulator needs to be melded into the operating system. This kit does as good a job of mitigating those difficulties as any that I have encountered. ■

MICROCOMPUTERS AND INTERFACES

We have six single board computers, two video boards and interfaces for the IBM-PC and APPLE computers. You can use our products for security systems, heat control, light control, automated slide show, irrigation systems, telephone systems, automated pool control just to name a few. Available. For catalog call or write to:

JOHN BELL ENGINEERING, INC.
400 OXFORD WAY
BELMONT, CA 94002
(415) 592-8411

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A COPY OF THIS WARRANTY IS AVAILABLE FREE, ON REQUEST.

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

DISK DRIVE SPECIAL!

5¼" Double Sided-Double Density

Brand New, Unused. Mfg. in Japan by Canon. 2/3 Height - Direct Drive! Industry Standard Pin Out. 6MS Access. 40 Tracks.

$49.95 ea | **2 FOR $85**

Sold on a First Come Basis
Add $2 Each U.P.S.

64K S100 STATIC RAM

$139.00 KIT

NEW!

LOW POWER!

150 NS ADD $10

BLANK PC BOARD WITH DOCUMENTATION $49.95

SUPPORT ICs + CAPS $17.50

FULL SOCKET SET $14.50

FULLY SUPPORTS THE NEW IEEE 696 S100 STANDARD (AS PROPOSED)

FOR 56K KIT $125

ASSEMBLED AND TESTED ADD $50

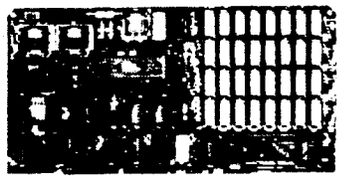
FEATURES: ***PRICE CUT!***

- Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- Fully supports IEEE 696 24 BIT Extended Addressing.
- 64K draws only approximately 500 MA.
- 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- 2716 EPROMs may be installed in any of top 48K.
- Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- BOARD may be partially populated as 56K.

256K S-100 SOLID STATE DISK SIMULATOR!

WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



BLANK PCB (WITH CP/M* 2.2 PATCHES AND INSTALL PROGRAM ON DISKETTE) $69.95 (8203-1 INTEL $29.95)

FEATURES:

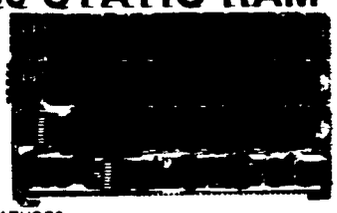
- 256K on board, using + 5V 64K DRAMS.
- Uses new Intel 8203-1 LSI Memory Controller.
- Requires only 4 Dip Switch Selectable I/O Ports.
- Runs on 8080 or Z80 S100 machines
- Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
- Provisions for Battery back-up.
- Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
- The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
- Compare our price! You could pay up to 3 times as much for similar boards.

$199.00

#LS-100 (FULL 256K KIT)

64K SS-50 STATIC RAM

$119.00 (48K KIT)



NEW!

LOW POWER!

RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $52

SUPPORT ICs + CAPS $18.00

FULL SOCKET SET $15.00

56K Kit $129

64K Kit $139

ASSEMBLED AND TESTED ADD $50

FEATURES:

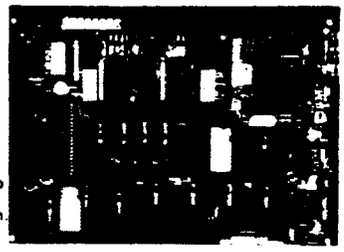
- Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- Fully supports Extended Addressing.
- 64K draws only approximately 500 MA.
- 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- Board is configured as 3-16K blocks and 8-2K blocks (within any 64K block) for maximum flexibility.
- 2716 EPROMs may be installed anywhere on Board.
- Top 16K may be disabled in 2K blocks to avoid any I/O conflicts.
- One Board supports both RAM and EPROM.
- RAM supports 2MHZ operation at no extra charge!
- Board may be partially populated in 16K increments.

THE NEW ZRT-80 CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:

- Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- RS232 at 16 BAUD Rates from 75 to 19,200.
- 24 x 80 standard format (60 Hz).
- Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
- Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- Composite or Split Video.
- Any polarity of video or sync.
- Inverse Video Capability.
- Small Size: 6.5 x 9 inches.
- Upper & lower case with descenders.
- 7 x 9 Character Matrix.
- Requires Par. ASCII keyboard.



BLANK PCB WITH 2716 CHAR. ROM, 2732 MON. ROM $49.95

SOURCE DISKETTE - ADD $10

SET OF 2 CRYSTALS - ADD $7.50

WITH 8 IN. SOURCE DISK! (CP/M COMPATIBLE)

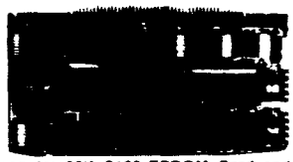
$99.95 # ZRT-80 (COMPLETE KIT, 2K VIDEO RAM)

32K S100 EPROM/STATIC RAM

NEW! FOUR FUNCTION BOARD! **NEW!**

EPROM II FULL EPROM KIT $69.95

A&T EPROM ADD $35.00



BLANK PC BOARD WITH DATA $30.95

SUPPORT IC'S PLUS CAPS $16

FULL SOCKET SET $15

We took our very popular 32K S100 EPROM Card and added additional logic to create a more versatile EPROM/RAM Board.

FEATURES:

- This one board can be used in any one of four ways:
  - A. As a 32K 2716 EPROM Board
  - B. As a 32K 2732 EPROM Board (Using Every Other Socket)
  - C. As a mixed 32K 2716 EPROM/2K x 8 RAM Board
  - D. As a 32K Static RAM Board
- Uses New 2K x 8 (TMM2016 or HM6116, RAM's
- Fully Supports IEEE 696 Buss Standard (As Proposed)
- Supports 24 Bit Extended Adressing
- 200 NS (FAST!) RAM'S are standard on the RAM Kit
- Supports both Cromemco and North Star Bank Select
- Supports Phantom
- On Board wait State Generator
- Every 2K Block may be disabled
- Addressed as two seperate 16K Blocks on any 64K Boundary
- Perfect for MP/M* Systems
- RAM Kit is very low power (300 MA typical)

PRICES SLASHED!

32K STATIC RAM KIT — $109.95

For RAM Kit A&T — Add $40

Digital Research Computers

P.O. BOX 461565 • GARLAND, TEXAS 75046 • (214) 225-2309

TERMS: Add $3.00 postage We pay balance. Orders under $15 add 75¢ handling No C.O.D. We accept Visa and MasterCharge Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50. add 85¢ for for insurance

*TM OF DIGITAL RESEARCH INC. (CALIF.)

WE ARE NOT ASSOCIATED WITH DIGITAL RESEARCH INC. (CALIF.) THE SUPPLIERS OF CPM SOFTWARE

BASE

A Series on How To Design and Write Your Own Database

By E.G. Brooner

In this issue we want to get away from the program itself, and take up a couple of subjects that are directly related.

Acquiring Remote Data

Our editor/publisher, who likes the idea of using micros for such things as process control, asked me a question for which I had no ready answer. It was, to summarize, "Why can't a database program acquire data, such as from remote sensors, without the intervention of a human operator?" As a diehard business programmer, I had never given it any thought; in my world everything goes in through the keyboard.

Nevertheless, I promised to consider the matter, especially as to how it could be related to this series on writing your own database software. I have not as yet tried any such thing, but my theoretical answer follows.

In an earlier column we defined the functions of database software. Two of those were the design of the data file structure and the entering of data; then came data retrieval. The final operation, of course, is the production of useful reports. It seems to me that the only place where remote data acquisition could 'fit in' was in the 'enter data' phase. (I can't imagine remote sensors designing a database structure.) You would naturally enter your design parameters, as described earlier, from the keyboard. At that point you would not have to be concerned with how the data would be input, but only with the form it would take. It would most likely be in the form of numbers or strings of some kind, which in the interest of usefulness, would have to follow some kind of fixed format. You, then, would design your records and files to recognize the incoming format and store that kind of data. Retrieval, and probably report printing, are still functions to be done at the machine itself.

The only trick would be getting the data (which might come at any time, completely unattended by a human operator) to write itself into the files which you have designed for it. To make this as simple as possible I envision it bypassing the 'normal' keyboard entry routine. The entry routine could be made a module of your overall database program that could be chosen from the main menu, or it could be a separate program which simply has the ability to access the relevant file(s). Either way, it could be added to what we have considered so far by simply adding another menu choice (something like 'Acquire remote data') which would chain the necessary input routine. Ah, the benefits of semi-structured programming!

Because we are talking about C-BASIC in this series, I tend to see this as a separate BASIC program which has the ability to read a port and write files. The file-writing code could be fixed for a particular operation. That is, if your base for acquisition is to have two data fields, that's all the little program need consider. It would of course have to be 'Run' by a human operator, who could then walk away and leave it to do its own thing. Its 'own thing' would be to sit there in a 'read loop,' checking the status of the port from which data might be expected to materialize. The program would detect a 'ready' status when data is present (as it would with any modem or terminal port), read the data, write it in the next empty record, and go back into a wait-state. In other words, the whole operation would not be much more complicated than the input routine that connects your terminal to your computer.

The final step would be the retrieval and printing of useful output. If you merely intend to acquire tables of data for a later printout, the print section of your database program could handle it easily. If you want it plotted, you would need a plotting program capable of reading your data files. (See descriptions of plotting programs in earlier issues of *The Computer Journal.*) If you wanted the data transmitted somewhere else for some purpose, there would be another routine to read only the affected files and handle that little chore. These routines could be incorporated into your homemade data base, or written as separate programs in any compatible language including assembler. The data could enter the system from a serial or parallel port via hard wiring or a modem — in other words, by the same means anything else might be connected.

In summary, the acquisition of random data, and its transmission to another point, should be considered as simply another peripheral operation that can be executed with any database system whose files are individually accessible.

The program BASE, even in its simplest form, should be suited to this kind of extra activity. Because it is a training exercise, the various sections have been kept simple and straightforward. The data files are in text form and can be written to or read by almost any CP/M compatible language. Its use does not depend on any special function keys or interlocking routines. It should only be necessary to make sure that all relevant files are updated by the remote operation; that is, all fields of a given record should be written (by blanks if necessary) and the 'bookkeeping' details such as number of records per file included each time data is written. Any necessary sorting (key files) could be accomplished later by the operator in the manner provided by the main program.

Why is BASE Menu-Driven Rather Than Form-Oriented?

Another database feature that we have been questioned about is why the entire program has been menu-driven. Many slick little databases let you design a 'form' on the screen, which is then filled in; the same concept applies

to printed output. Again, I have to admit that BASE is a training exercise which we hope to see applied to almost any conceivable CP/M configuration, allowing it to be terminal and printer independent. All of those fancy features depend on direct cursor or printhead control, which varies from terminal to terminal and among different printers. Also, some programs do cute things by calling internal routines; and again, we opted for maximum portability. In BASE the menus fit the screen and the only control built into the program is a 'clear screen' command. It is a subroutine (line 9999) which you only have to change once to adapt the program to different terminal configurations. And all I/O uses standard CP/M calls.

If you know how to address your terminal's cursor it is not difficult to rewrite the entry routine to present an on-screen form rather than a menu.

Different Versions of C-BASIC

A note should be made, too, of the fact that CBASIC can be compiled in two ways—by using the CBASIC and CRUN packages that come with the cheaper version, and by using CB-80, which is a true compiler that is not always available to the low-budget hacker.

A few minor differences creep into the code, which in either case is written by a text editor. These will be accounted for in REMARK statements as the program is presented. Needless to say, if you write the code for CB-80 it will not work properly with CRUN and vice versa. Both versions require, in addition to program space, room for some arrays which can be quite large depending on the data you put in them, and for sorting. 'Modern' micros are hardly memory-poor, but many of us hackers are still fighting to stay within 48K or less and have to watch these things carefully.

It should also be noted in passing that the CB-80 version, which is in true

(continued on page 27)

MODEL 100 C COMPILER

Now you can write efficient programs for your TRS-80 model 100 with ease. Or, learn the essentials of C programming while traveling!

C/100 - THE "PORTABLE" C COMPILER
Cassette version $49.00
Disk/Video interface version $59.00
Model II version (run on mod II, then download object code to model 100) $79.00
Model III version (as above for Mod III) $79.00

Write or call for information on other TRS-80 software.

MODELS II, 12, 16
MODELS III, 4

TRS/C C COMPILER
Full K&R with source to the function library. UNIX compatible $85.00

ZSPF EDITOR
SPF, the choice of most mainframe programmers, is now available for Z80 machines. And it's panel driven so you can customize it! $75.00

business utility software
109 minna ste 423 san francisco ca 94105
(415) 397-2000

APROTEK 1000™ EPROM PROGRAMMER

only $250.00

TECHNICAL BREAKTHROUGH NOW ALLOWS A PRICE BREAKTHROUGH

A SIMPLE, INEXPENSIVE SOLUTION TO PROGRAMMING EPROMS

The ***APROTEK 1000*** can program 5 volt, 25XX series through 2564, 27XX series through 27256 and 68XX devices plus any CMOS versions of the above types. Included with each programmer is a personality module of your choice (others are only $10.00 ea. when purchased with ***APROTEK 1000***). Later, you may require future modules at only $15.00 ea., postage paid. Available personality modules: PM2716, PM2732, PM2732A, PM2764, PM2764A, PM27128, PM27256, PM2532, PM2564, PM68764 (includes 68766). (Please specify modules by these numbers)

APROTEK 1000 comes complete with a menu driven BASIC driver programmer listing which allows READ, WRITE, COPY and VERIFY with Checksum. Easily adapted for use with IBM, Apple, Kaypro, and other microcomputers with a RS-232 port. Also included is a menu driven CPM assembly language driver listing with Z-80 (DART) and 8080 (8251) I/O port examples. Interface is a simple 3-wire RS-232C with a female DB-25 connector. A handshake character is sent by the programmer after programming each byte. The interface is switch selectable at the following 6 baud rates: 300, 1.2k, 2.4k, 4.8k, 9.6k and 19.2k baud. Data format for programming is "absolute code" (i.e., it will program exactly what it is sent starting at EPROM address 0). Other standard downloading formats are easily converted to absolute (object) code.

The ***APROTEK 1000*** is truly universal. It comes standard at 117 VAC 50-60 HZ and may be internally jumpered for 220-240 VAC 50-60 AZ. FCC verification (CLASS B) has been obtained for the ***APROTEK 1000***.

APROTEK 1000 is covered by a 1 year parts and labor warranty.

FINALLY — A Simple, Inexpensive Solution To Erasing EPROMS

APROTEK-200™ EPROM ERASER
Simply insert one or two EPROMS and switch ON. In about 10 minutes, you switch OFF and are ready to reprogram.
APROTEK-200™ only $45.00.

APROTEK-300™ only $60.00
This eraser is identical to ***APROTEK 200™*** but has a built-in timer so that the ultraviolet lamp automatically turns off in 10 minutes, eliminating any risk of overexposure damage to your EPROMS.
APROTEK-300™ only $60.00.

APROPOS TECHNOLOGY
1071-A Avenida Acaso, Camarillo, CA 93010
CALL OUR TOLL FREE ORDER LINES TODAY.
1-(800) 962-5800 USA or 1-(800) 962-3800 CALIFORNIA
TECHNICAL INFORMATION: 1-(805) 482-3604
Add Shipping Per Item: $3.00 Cont. U.S. $6.00 CAN, Mexico, HI, AK, UPS Blue

DSD 80
FULL SCREEN SYMBOLIC DEBUGGER

"THE SINGLE BEST DEBUGGER FOR CP/M-80. A TRULY AMAZING PRODUCT."
[illegible] FOR PCW MAGAZINE AUTHOR OF BOX...

Complete upward compatibility with DDT
Simultaneous instruction register stack & memory displays
Software In Circuit Emulator provides write protected memory, execute only code and stack protection
Full Z80 support with Intel or Zilog Mnemonics
Thirty day money back guarantee
On line help & 50 page user manual

NOW ONLY $125.

SOFTADVANCES
P.O. BOX 49473 AUSTIN, TEXAS 78765 (512) 478-4763

Interfacing Tips and Troubles

A Column by Neil Bungard

In the next two installments of "Interfacing Tips and Troubles," we will look at a device which will allow you to communicate to a computer over telephone lines without the need to carry a modem or a terminal. This device, the *TONE CONTROL*, is installed in parallel with the telephone and interfaces directly to the computer to allow the computer to accept touch tone telephone signals as input data. With the TONE CONTROL, data can be relayed, telephone numbers can be recorded, or devices can be controlled in your home from the other end of the country.

The TONE CONTROL was originally created for a legal firm as part of an automatic billing system. This system monitored the telephone numbers of outgoing long distance calls, recorded the time, date, and duration of each call, then automatically billed a lawyer's clients for the calls that were made. Over the years, I have used the TONE CONTROL circuit in other projects as well, and thought that you might find it interesting.

The TONE CONTROL was designed as a general interface circuit, which means that I have not specified or described its use with any particular computer. If you have an interest in building this device and feel that you need more information to interface it to your specific machine, drop me a line here at *The Computer Journal* and I will be glad to help in any way that I can.

Some Touch Tone Telephone Basics

Before going into an explanation of the TONE CONTROL circuit, let's look at a general description of what the TONE CONTROL does, and review some touch tone telephone basics. TONE CONTROL's function is to determine if any one of the 12 keys on an incalling touch tone telephone is being pressed, and if so, identifies the key. When a key is pressed, the tone control alerts the computer and sends it a 6 bit code which identifies the key being pressed. In addition, the "*" and the "#" keys on the touch tone keypad can act as double function keys to allow the user to specify the keypad's alpha character set.

TONE CONTROL identifies keypresses in the same manner that the telephone company does. Looking at the telephone, notice that the keypad is configured in 3 columns and 4 rows. A unique frequency is assigned to each row and each column so that when a single key is pressed, the frequencies for both the row and the column are present on the line simultaneously. When two frequencies are mixed in this fashion, a principle called *heterodyning* occurs. This principle states that when two frequencies are simultaneously present on the same line, the two "fundamental" frequencies and their sums and their differences will also be present. This means that careful selection of the row and column frequencies is necessary to ensure that the sum and difference frequencies will never be the same as any row or column frequency. Bell telephone has carefully selected the row and column frequencies so that no conflicts occor; these frequencies are shown in Figure 1.

Circuit Description

Referring to TONE CONTROL's block diagram, (Figure 2), TONE CONTROL contains 7 phase locked loops.

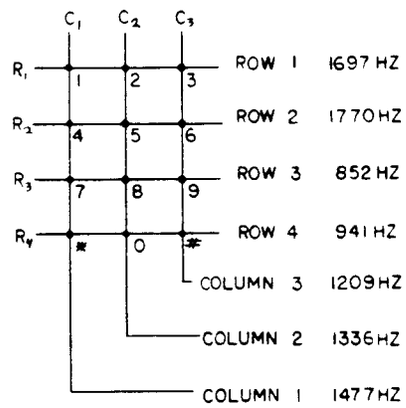


Figure 1

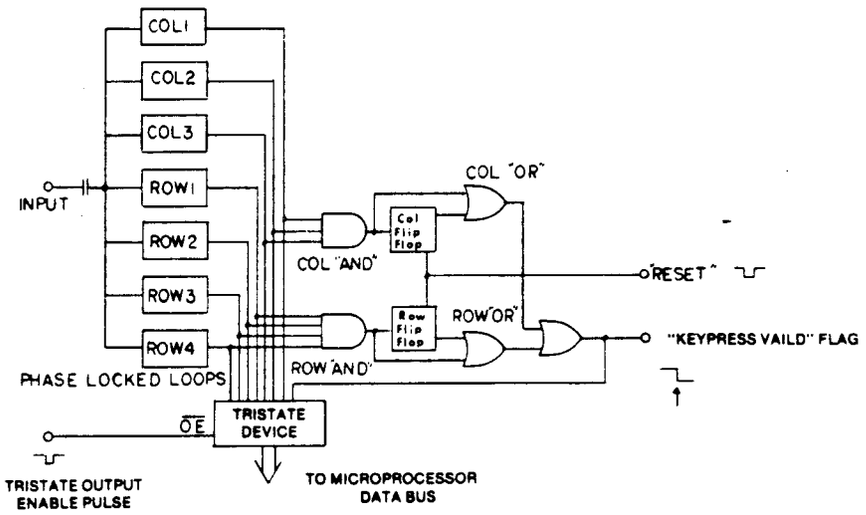


Figure 2: Tone Control Block Diagram.

Each phase locked loop (PLL) monitors the incoming signal, and if the signal is the same frequency that the PLL has been tuned to detect, it sets its output to a logic "0." If the input signal is not the frequency that the PLL has been tuned to detect, its output will be set to a logic "1." Each PLL is tuned to a separate row or column frequency, and since there are three unique column frequencies and four unique row frequencies, seven PLLs are needed. Referring to Figure 2, if a single touch tone key is pressed, the outputs of one row and one column PLL will go to a logic "0." This will cause the outputs of the row and column "AND" gates to go to a logic "0" which forces the row and column flip flops, and the row and column "OR" gate outputs to a logic "0" as well. The row and column "OR" gate outputs are again ORed and the output of this final gate is used to alert the computer that a valid keypress is available. The function of the final "OR" gate is to ensure that both row and column signals are present simultaneously. This is important because voice or other noise on the line can trigger individual PLLs. However, the chances are remote that voice or other noises will effect both row and column PLLs simultaneously, so this final "OR" minimizes false keypresses.

When a valid "dual tone" signal occurs, the TONE CONTROL alerts the computer that a valid keypress is available. The keypress information is present on the input of the octal tristate device. With the output of the octal tristate device connected to the data bus of the computer, all that is required to get information into the computer is to strobe the OE input on the octal tristate device to a logic "0." Since the processor reacts much faster than a key can be released, it will be necessary for the computer to reset the "keypress valid" flag as soon as it has the keypress information. If this signal is not removed, the processor may input the same keypress hundreds of times before the key can be released. The "keypress valid" flag is set to a logic "1" by strobing the "reset" input on the row and column flip flops with a low going pulse. Ideally, this would be the last operation that would be performed in the "input keypress" subroutine.

Next month we will present the TONE CONTROL circuit, and talk about how it is constructed and tuned to detect the individual dual tone frequencies. We will also look at how this device is used for typical application. ■

Classified

The Computer Journal will carry Classified Ads. The rate is $.25 per word. All Classified Ads must be paid in advance, and will be published in the next available issue. No checking copies or proofs are supplied, so please type your ad or print legibly.

Book Sale—These books are offered at this price while the supply lasts.
Zilog Z80-CPU Technical Manual, $1.50
The Programmer's CP/M Handbook
by Andy Johnson-Laird, $18.66
Real Time Programming—Neglected Topics
by Caxton C. Foster, $8.46
Microprocessors for Measurement and Control
by D.M. Auslander and P. Sagues, $14.41
CBASIC Users Guide
by Osborne, Eubanks, and McNiff, $15.26
Introduction to FORTH
by Ken Knecht, $9.31
FORTH Programming
by Leo J. Scanlon, $14.41
Interfacing and Scientific Data Communications and Experiments
by Peter R. Rony, $6.76.
These prices are postpaid in the U.S. only. TCJ, PO Box 1697, Kalispell, MT 59901.

Morrow Decision I S-100 system with MPZ-80 CPU, DJ/DMA floppy disk controller, 256K static ram, Wonderbus I/O on mother board, Disk Jockey Hard Disk (HDCA) Controller, 801 floppy drive, 10MB hard disk, CP , Micronix Multiuser system, unconfigured MP II, dBase II, Wordstar, Accounting Plus. Excellent condition. $3500, some trades considered. TCJ, PO Box 1697, Kalispell, MT 59901 (406)257-9119 days, or (406)755-4654 evenings.

Corvus 10MB Hard Disk for the Apple II plus, $995; Videx 80 column board for the Apple II plus, $200; Apple Numeric Keypad, $75; Miscellaneous Apple III software at half price. All plus shipping. Cashier's check or money order. The Computer Place, 36 2nd Street East, Kalispell, MT 59901, phone (406)755323.

The Security Disk. Protected vs unprotected. At last, the best of both worlds. Two hadrcore programmers team up to give you software designed to protect you private files, plus the ability to analyze and copy your other "Locked-up" disks to standard format. Simple "Password" protection to "Cryptology". A disk packed with secrets, tips, and other goodies. Not locked-up, listable, and modifiable. Completely REM-arked. To order send $24.95 CHECK/MO to B.M.E. Enterprises, Box 191, Kila, MT 59920.

THE COMPUTER CORNER

A Column by Bill Kibler

Well here goes another month worth of ramblings, and do I have some great news this month. My first issue of our club newsletter was a big hit, but that's not half of what's going on. If you live in central California or near it, mark the calendar for May 4. Our club is older than *Byte* magazine and so we get to beat them to celebrating the 10th year of microcomputing. How about that, microcomputers have been around for TEN YEARS!!!

For the user, this year will be packed with festivals and parties to commemorate this big event. The Sacramento Microcomputer Club will be having a celebration on May 4, with some big name guest speakers. So far, Godbout is a confirmed speaker and he's promising to bring George Morrow too.

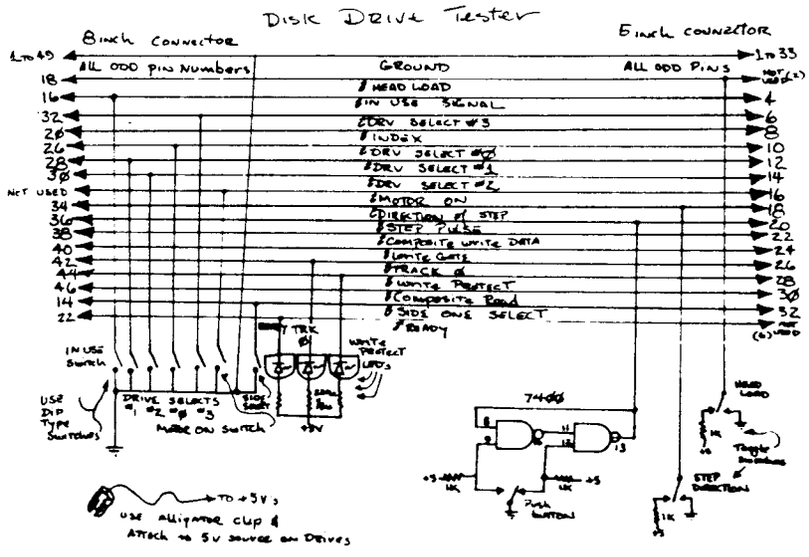
Despite all this mania for the tenth year, some hacking and modifying has still been going on. I just discovered that the reason a systems disk would not work on any other system was that the MPI drives were two tracks off. I didn't believe it at first, but I discovered the track numbers were always two off when I was trying to set up another system and doing full track reads. So out came my alignment disk and my adapter for testing drives (see Figure 1) and sure enough the drives were two tracks off. After a little tweaking I can now read the disk on other systems, so take notice that not all of the incompatiblity problems between mini drives are in the software.

Talking about disk problems, my newest system is an ATR8000 (from SWP Microcomputer Products, Inc., 2500 E Randol Mill Rd, -125, Arlington, TX 76011, (817) 924-7759). Those of you who own Ataris may have heard of this system. SWP is a little know company with a slick system for interfacing Ataris to regular disks and other more normal peripherals (Atari does everything in serial). The best part of the unit is that the CP/M option can be used with a regular terminal in place of the Atari. A 1797 disk controller handles both 8" and 5" drives. Now this isn't much, since it only has 64K, but when treated as a single board system, it beats the Ampro because of the ability to handle 8" drives. In fact, that's how I look at it—just another single board unit for the same price as the others, but with better software. The unit has been able to read disks with different format changes in the middle of a track, something I haven't heard of before. The software also has a copy program for different formats, so this will become my media transfer system.

The ATR system got me poking around in a Kaypro trying to get a compatible system disk. I found out some rather interesting things. I will not go too deep here, but the system puts the new disk format information at D300 or so, and this is where you will need to poke different values for odd disk formats. I had the right formats but the wrong skew table. I never did get it right but I'll try more later, and besides it was lots of fun anyway. The biggest fault I find with most copy programs is that they do not handle the system track, or supply a sysgen function. Now normally that's not a big problem, but system changing is what I do the most. In fact, I am still rewriting a CCS 2422 BIOS as I have given up on using their regular BIOS for mini drives (it was intended to be my copy system).

SIG/M public domain disks are up to 217 volumes, although there will most likely be many more before you read this. One of the later disks has a convert program to move those 8080 code programs to 8086/88 code for assembling. In fact, that is what most of the big software houses did to change the code to 16 bit. This accounts for the slow and sometimes buggy programs that have found their way to IBM compatibles.

Speaking of IBM, the word is that the IBM PC is dead. It seems that IBM has a three year product life and the three years are up. What is next is a smaller version (about half the size) with 3½" drives. Of course there will be two models, one without any expansion and one with. Personally, I think this is going to upset some people's plans, but then what do you expect when the public jumps on a name and not a proven product.



If you can't guess, I do not love the IBM PCs, but their price may change that. With the demise of the IBM PC, prices will be coming down in big jumps. Soon you will be able to have a compatible with plently of expansion sockets and probably better speed than the IBM-PC for $500 to $800. My old Z100 is doing fine for my friend, and in fact he's been finding out all kinds of good stuff about it. One is why the machine is so slow, even at 8MHz. It seems that the screen characters are in RAM and therefore every character is changed into nine separate eight bit words that have to be written to the screen's memory for one screen character. That means my 4MHz Z80 could probably update a text screen four times faster than the Z100. However, the bit mapped graphics is for just that, graphics, not text, and that is where Zenith went wrong. They should have a standard character font mode which switches in a ROM and works like most systems, one memory location per letter. Lets see, now how does that chip work and where's my knife.....

Last month I talked about Packet Radio and it now seems that everywhere I turn there is another article about it. The latest issue of Forth Dimensions (FIG Forth newsletter) has an article on it, and I discovered that some back issues of MicroCornnucopia had a series. Hopefully next month I will have some more information on Packets and what little problems my many systems have thrown at me. ■

(Hacker Mac continued from page 9)

Accordingly, my concept of the minimal configuration Hacker's Mac is of a 68000 (possibly 68010) with 128K of RAM and a bitmapped display screen. It has a keyboard port, a mouse port, a bidirectional parallel port and a serial port. Its ROM defaults to terminal mode, but allows the user to download code from the serial port and execute it.

There are many unfilled portions of the board in this configuration. There are IC patterns for 16 more RAM chips, which allows 256K (using 64K chips) or 1 Mbyte (using 256K chips). There are connector patterns to allow a piggyback RAM board which doubles the capacity of the machine. There are a total of 28 pin ROM sockets with provision for piggyback ROM of equal size.

The video section has connector patterns which allow for an external display generator to provide addresses to the display RAM. The display RAM area itself can be extended off the board through these connectors.

The most important section is the DMA channel. It is not populated in the minimal version, but is contained on the PC board and can be implemented by loading components. It uses the 68450 4-channel DMA controller and drives a connector through which all four channels are made available outside the case. These channels may be broken out and individually latched by an external expander board, thus allowing simplified interfacing and most importantly, the ability to interface simply with as many as four other such ports.

This is the door through which the hardware architecture is made open. There is no hardware bus with physical connectors, but rather a structure of memory areas dedicated to the interchange of data with outside devices such as disc, coprocessors, virtual memory, LANs and external expansion chassis with their own busses.

What About Software?

It should be obvious by now that this is a software intensive design. Most of the work will have to go into the ROM and the utilities. Since I claim no expertise in these areas, I encourage a process of discussion, debate, factionalization, demonstration of better ways, reconciliation, and all the other things that go on in a viable society in regards to the software development.

I expect a similar process with regards to the hardware. I cannot hope to control the hardware except by being first with a good design. Even that is no guarantee, and I will have to let my designs stand or fall according to the critical evaluation of others. I know that I cannot do good hardware design without the involvement of software people as the design develops. That's why I am making this much fuss about it in advance instead of "just doing it." To proceed in isolation would produce a meaningless result.

There is no money behind this project. It is a dubious proposition, and will not attract investment in the near future. Sensible people have no business wasting time on it. It is a project for inveterate dreamers, for people who know exactly what they would do if given a chance to do it right, and for people who thought that they had that chance when the pickings were good but who found the rug pulled from under them when things got a little tight.

It's our project, if we want it bad enough.

Contact Information

Gordon French is making his bulletin board, The French Comnnection, available for this discussion. The number is (408) 736-6181, 1200 Baud. Use keyword SWIFT. This document will be available as a note on that board.

Within a month a hacker's section of the Whole Earth Electronic Library will be established on a network, although initially limited to access from area codes 415, 408 and 707. I have agreed to serve as a toastmaster on this board, which is scheduled to open in mid February.

I still intend to call a meeting at the exploratorium in about 30 days to allow interested hackers to meet and talk face-to-face. Once this document is published and I can assess its impact, I will go ahead with this meeting.

Epilog

I do not intend to be involved only as an organizer. I, like most of you, would much rather be designing than bringing other people together. There is room for many people to serve this organizing function, and I think we will all have to become organizers to some small extent. We shouldn't turn that function over to a special caste, or we'll regret it. ■

Contact Lee Felsenstein, at Golemics, Inc., 2600 10th Street, Berkeley, CA 94710, (415)486-8344 for further information or discussion.

New Products

C Language Seminar/Workshop

Computer Language magazine is sponsoring a C language seminar and workshop to be held September 16-18, 1985 at the Sheraton-Commander Hotel in Cambridge, Massachussets. Among the list of speakers confirmed for the event are: Jim Brodie, ANSI C Standards committee chairman; P.J. Plauger, co-author of *Elements of Programming Style* and ANSI C secretary; Leor Zolman, compiler writer and public domain C expert; Heinz Lycklama, chairman of the /usr/group UNIX Standards group; Robert Ward, coordinator of The C User's Group; Tom Plum, author of *Learning to Program in C*; and Larry Rosler, ANSI language subchairman.

For more information contact Beatrice Blatteis or Carl Landau, CL Publications, 131 Townsend St., San Francisco, CA 94107 phone (415)957-9353.

FORTH Literature at a Discount

The best in FORTH literature is now available at a 10% savings directly from the FORTH Interest Group (FIG). Edited and reviewed by FORTH experts, the recommended publications include tutorials, conference proceedings, listings, advanced topics, and more.

The FORTH Interest Group is a worldwide non-profit member supported organization with over 70 chapters and 5,000 members. FIG membership of $20 per year ($33 foreign) includes a subscription to *FORTH Dimensions*, the bi-monthly publication of the group. FIG also provides an on-line data base, a job registry and other services.

Call the FIG HOT LINE (415)962-8653 or write FIG, PO Box 8231, San Jose, CA 95155 for more information.

I/O Pro Development System

MEF has announced I/O PRO, which is a system of software development tools that create an enhanced environment for developing interactive FORTRAN and PASCAL programs for IBM compatible micros. I/O PRO includes a program for creation of text and graphics screens used as the input and output media for interactive programs. It also serves as a versatile presentation development and display program.

The I/O PRO Development System offers an alternative to the conversion of existing analysis codes in FORTRAN or PASCAL to C or BASIC to obtain a programming environment that will allow development of intelligent systems and applications such as data acquisition and control. Creation and editing of text and graphics is accomplished with easy to use word processing commands, and color and high resolution graphics, including animation, are supported by MEF's graphics library.

I/O PRO includes a screen development and testing module, plus a library module consisting of a FORTRAN/PASCAL screen caller, a screen conversion routine allowing existing screes to be converted to I/O PRO format and 92 utility routines that allow the FORTRAN/PASCAL programmer to fully utilize the capabilities of the IBM-PC and compatibles. A third module can be combined with the screen development module to create, edit, and show onscreen "slide presentations." Information and a demo/tutorial are available from Andy Montz at MEF Environmental, Inc., P.O. Box 26537, Austin, TX 78755, phone (512)251-5543.

cVIEW Screen Manager

CompuCraft has announced their cVIEW 2.11 screen management tool for IBM-PS's and compatibles. It allows software developers working in C to build sophisticated user interfaces and can be used with the Computer Innovations, Mark Williams, Microsoft or Lattice C compilers.

Input fields are defined as they are placed on the screen, and type specification of the input fields provides automatic testing and rejection of incorrect user entries. Minimum/maximum range limits may be specified for numeric fields, and user written edit routines may be applied to any field. All of the special keys can be defined by the programmer for each form.

The cVIEW package contains the editor used to create and modify forms, online help for each level of operation, the runtime library required for interfacing to applications programs, and programmers reference manual. A demo disk containing the cVIEW editor and online help files is available for $25 which is credited to the purchase of cVIEW. For further information contact Jerry Januzzi at CompuCraft Corp., 42101 Mound Road, Sterling Heights, MI 48078, phone (313)731-2780.

Free Data Acquisition & Control Handbook

MetraByte is offering a free 68 page handbook describing their data acquisition and control plug-in boards for the IBM-PC and Apple computers. The handbook includes sections on applications, configuration guides, example programs, accessories, hardware, and software cross reference charts.

Contact Robert J. Rogers at MetraByte Corporation, 254 Tosca Drive, Stoughton, MA 02072 phone (617)344-1990 for your copy.

Computer Biofeedback System

Rosetronix has announced two biofeedback systems. They claim that the brain wave system, which interfaces to the ZX-81/TS-1000 and Apple computers, permits the user to control a programmed English vocabulary. The print out can be to screen, printer, or synthesizer and permits brain wave control over any external robotic device with 256 different commands. The galvanic skin response system prints out percentages of stress with English sentences and can also be attached to a printer, synthesizer or robotic device. They state that it can be used as a talking lie detector. Contact Rosetronix at PO Box 7464, Chula Vista, CA 92012 for more information.

Books of Interest

The Self-Publishing Manual

by Dan Poynter
Published by Para Publishing
Post Office Box 4232
Santa Barbara, CA 93103-0232
348 pages, 5½ × 8½", $14.95

The author, Dan Poynter, fell into self-publishing after he researched and wrote a technical book on the parachute. Realizing that no publisher would be interested in this book, he had the book printed and published it himself. The book sold well, and Dan went on to write and publish additional books.

Publishing a book is a very involved process where many things all have to be done correctly and at the right time. It can be especially difficult when the book is your own, because there is no one to look over your shoulder and temper your decisions. It can also be especially rewarding, because no one else understands your book and its market as well as you, the author.

This book starts by explaining the publishing options, tells you about writing your book and starting your own publishing company, advises you about dealing with the printer and promoting your book, and how to deal with customers and dealers. The Self-Publishing Manual is about the **business** of writing and publishing, not merely an author's guide. The contents of the book are as follows:

•**Chapter 1 Your Publishing Options and Why You Must Self-Publish.** The difference between authors, printers, and publishers; the book publishing industry; the big publishing firms and why books don't stay in print; royalties; small presses; vanity or subsidy publishers; literary agents; self-publishing; and book printers.
•**Chapter 2 Writing Your Book — How To Generate Saleable Material.** Picking a subject; writers block; time; write your ad before your book; the flow of your manuscript; writing style; copyright and contracts; parts of a book; typesetting; artwork and preparation for printing
•**Chapter 3 Starting Your Own Publishing Company.** Forms of businesses, sole proprietorship, partnership, or corporation; publications you should read; your company name; your place of business; setting up your office; legal aspects; raising money; and making a profit.
•**Chapter 4 Printing Your Book. Materials — Design — Printing.** Printing processes; book design; standardize and save money; number of pages and typefitting; illustrations; paper; paperback or hardcover; the cover of your book; binding; typesetting; and printing.
•**Chapter 5 Announcing Your New Book — Getting Listed.** International Standard Book Number; Universal Product Code; Cataloging In Publication; Library of Congress; Copyright; The Cumulative Book Index; and directories.
•**Chapter 6 Prices, Discounts, Terms, Collections, and Returns.** Establishing the retail price; how many should you print? estimating sales; reprints; discounts, and returns.
•**Chapter 7 Promoting Your Book.** Getting free publicity; analyze your market; rate of sale; promotion is up to you; coding your list; promotion services; testimonials; news releases; reviews; selecting magazines; promotion through articles; and seminars.
•**Chapter 8 Who Will Buy Your Book? Finding Customers.** Size of the industry; audience specialization; repetitive audience contact; multiple markets; seasons; wholesalers and distributors; bookstores; libraries; non-book outlets; rights; and book fairs.
•**Chapter 9 Selling Your Book — Advertising.** Advertising basics; creating ad copy; advertising agencies; your brochure; direct mail advertising; mailing lists; circulars; the direct mail letter; magazine advertising; and remainders.
•**Chapter 10 Distribution, Getting Your Book to Market.** Mail order; The Federal Trade Commission; order processing; inventory control; and shipping.
•**Chapter 11 Computers and Book Publishing.** Word processors and writing; computerized typesetting; fulfillment; promotion; market research; and financial management.
•**Chapter 12 Coping With Being a Published Author.** Your new status; articles; consulting; and speaking.
•**Appendix A Your Book's Calendar.** A 44 item check list of what to do when.
•**Appendix B Resources.** A well organized 17 page list of sources for information.
•**Index.** There is a large index which makes it easy to locate the items you want.

This is not a theoretical textbook on publishing, but rather a step-by-step manual describing the mechanics of publishing your book. The publishing industry has some rather strange business practices, especially relating to terms and return policies, and reading this book can prevent a disastrous surprise.

The subject of this book is different than that of the technical books which we normally review, and you may wonder what self-publishing has do to with computers. Because of the current state of the technical publishing industry, the only way that specialized technical books will be published is if the author publishes the book himself. The information in this manual can help make your book a success.

I recommend the book to anyone considering publishing any book, booklet, or manual, or anyone who is interested in the publishing industry. If I was still running my print shop I would make this manual required reading for anyone who wanted me to print their book. ■

Advertiser's Index

Apropos 19
Barnes Research 26
John Bell 16
Bersearch 9
Business Utility Software 19
BV Engineering 5
Computer Trader 12
Digital Research Computers 17
Echelon, Inc 26
Micro Methods, Inc 12
Miller Microcomputer Services 7
Poor Person Software 12
Public Domain Software 12
Rio Grande Robotics 27
Soft Advances 19

"BMON"

Software In-Circuit Emulator

Links your CP/M computer with any Z80 based computer or controller that you may develop. All that is needed is BMON, 8K of ROM space, and a handshakeable bi-directable I/O port (either RS232 or Parallel).

Features:

—Full program development debugger with Breakpoints, Snaps, Stops, & Waits.

—Single Step program execution.

—Download file from CP/M system to development RAM.

—Upload Memory from development RAM to CP/M disk.

—Two versions: Master BMON runs in your CP/M system, Slave BMON runs in your target system.

Note: Requires Microsoft's M80 & L80 assembler & linker to setup Slave BMON.

8" SSSD Disk containing Master BMON, Slave BMON, CONSOL, BMONIO, CONSOLIO, and Users Manual $49.95

Shipped Via prepaid UPS
—No COD or P.O. Box—
Check or Money Order to:

Barnes Research & Development
750 W. Ventura St.
Altadena, CA 91101
(818) 794-1244

CP/M is a trademark of Digital Research Inc
M80 & L80 are trademarks of Microsoft Inc

Z sets you FREE!

Z — yes! Synergistic combination of ZCPR3 and ZRDOS2 produces flexible state-of-the-art Z80 operating system with tremendous productivity features.

Z-System consists of software modules, dynamic loading segments, and tools permitting optimum computer usage ranging from production program development to turnkey, password-controlled, end-user installations. Facilities include: multiple commands per line, file search paths, named directories, I/O redirection, command flow control, screen-oriented menu generators, complete housekeeping file and directory management, shells, alias (scripts) and nested-alias generation, and complete online help.

Seventy-six support utilities, five tool packages, and two application programs available now! Fully upward compatible with CP/M-80.

Z can now be purchased as auto-install program (Z-Com) or as manual-install ZCPR3 with semi-auto install ZRDOS package (Z-System). Our latest versions, to be released this year, support Zilog Z800 and Hitachi HD62801/64180 high-technology chips, chips run existing 8080 and Z80 programs!

Echelon eight-bit operating systems written in Assembly Language, using linkable macro subroutine libraries, offer performance paralleling best single-user 16/32-bit microcomputer systems.

1. **Z-Com** Full-up Z Operating System with input/output redirection running under CP/M-80, online command and utility documentation and help system **$219.95**

2. **Z-System** Manual-install ZCPR3 and ZRDOS2, easily tailored by programmer to custom needs; source code to core and utilities; similar to Item 1 . . . . . **$199.95**

3. **Z-Tools** Four software development system packages permitting advanced, structured program design, macro relocating assembler, linking loader, librarian, cross-reference generator, debugger, mnemonic and pseudo-op translators, and interactive disassembler. Super $315.00 package value **$200.00**

4. **DSD** Dynamic Screen Debugger offers high-level features never before found in microcomputers; simultaneous display of dual-memory segments, stack, cpu states, and flags, with software In-Circuit-Emulation **$149.00**

5. **The Libraries** Linkable ZCPR3 libraries (Vlib, Z3lib, and Syslib3) of over 400 subroutines used for Assembly Language program writing. Simplifies structured, efficient code production; online help system and full source code provided **$45.00**
Syslib3 alone **$29.00**

6. **Term3** New generation communication program permits menu control of computer/modem operations between operator and time-share services, bulletin-boards and other remote computer systems; auto-answer to command-line prompt **$99.00**

7. **Discat** Fancy file and disk catalog program running under Z-System, menu driven and easily customized by operator **$49.00**

Fortnighter newsletter, 24-hour BBS Z-Node System keep Z users informed of microcomputer happenings. Write or call for brochure or order now! State disk format desired; add $3.00 shipping & handling; Californians please add 6-1/2% sales tax. Visa/MC, check, money or purchase order accepted. (Program names are trademarks of their respective owners.)

Echelon, Inc.

101 First Street • Los Altos, California 94022 • 415/948-3820

Editor *(continued from page 1)*

puterists, especially those doing experimental work, should be exposed to what is going on with systems other than the one that they are currently using.

More Articles Needed

We are finally getting enough articles to expand the magazine and to plan a month or two into the future; but we still need more good technical articles, especially on general subjects such as disk drive controllers, A/D and D/A, the IEEE-488 interface, robotics, UARTS, and control applications. The majority of the submitted articles have been based on the Apple II and the CP/M S-100 systems, apparently because the ease with which cards can be added to their open bus makes them a hardware hacker's delight. We are looking for projects designed to interface through a standard Centronics or RS-232 port so that they can be used with various computers, but most of the work is still being done by the people with an open bus system who design for their specific system. We receive requests for more articles on the Atari, C-64, TS-1000, and other systems, but very few people working with those systems have submitted articles. ■

(Graphic listing continued from page 15)

```
          ROL.W    #8,D2
          MOVE.B   D2,VIDLO
          MOVE.B   VIDIO,D1          *Get the byte
          BSR      WRTLST            *Send to list device
          ADD.W    #80,D2            *Go to next video line
          CMP.W    #$9600,D2         *Test for end of screen
          BCS      LINE1
          SUB.W    #$9600,D2
          MOVEQ    #$0D,D1
WRTLST:   MOVEQ    #5,D0             *Write list function
BDOS:     MOVE.L   D2,-(SP)          *Save pointer
          TRAP     #2                *Invoke BDOS
          MOVE.L   (SP)+,D2          *Restore pointer
          RTS
*
          DATA
*
VSET      DC.B     $0D,$1B,'A',0,0 *Return and set v.s. string
GSET      DC.B     $1B,'K',224,1,0 *Set graphics mode string
NSET      DC.B     $1B,'2',$0C,0   *Set normal and f.f. string
*
          END
```

(Base continued from page 19)

machine language (CP/M .COM file) runs a great deal faster. In fact, it lets us use (in the code to be presented) a very simple sort routine that might not otherwise do for large data files. It is, in fact, the much-maligned bubble sort.

We'll go into that section in the next article, when we learn how to assign, sort, and use the key files. And if you are feeling creative by then, the completely separate sorting program is the place to implement any fancy sort/search ideas that you may have. The data files will already be there if you have followed the previous explanations and listings, and they are adaptable to a wide variety of special treatments. ■

PERSONAL ROBOTICS EXCLUSIVELY
FACTORY AUTHORIZED DEALERS

HAVE YOU HEARD ABOUT SAVVY?

SAVVY is a revolutionary programming language for the Apple II+ & IIe and the IBM PC. It approaches artificial intelligence. Besides being a perfect language for robot control, it can handle general applications programming. Write or call for detailed descriptive material.

A FULL-LINE ROBOT SUPPLIER

1. No dissatisfied customers.
2. Call if you have questions—we know our products.
3. Unlimited help when you need it.

HEATH	List	SALE
MOVIT		
Screwdriver Kits—No Solder—FUN		
Avoider	44.95	40.15
Circular	67.95	63.75
Line Tracer	39.95	36.75
Medusa	27.95	26.25
Memoconcrawler	74.95	70.15
Monkey	24.95	23.65
Mr. Bootsman	30.95	29.25
Peppy	24.95	23.65
Piper Mouse	44.95	40.13
Sound Skipper	24.95	23.65
Turn Backer	39.95	36.75
We have sold hundreds of these items. They work as they should and they last.		
HARVARD ASSOC.		
Turtle Tot	299.00	275.00
Excellent for simple educational applications. Our customers have been very satisfied.		
ROBOT SHOP		
Robot Bug	129.95	122.00
Z-1	149.95	136.50
Z-2	249.95	223.50
Simple and dumb, but lots of potential for customizing and expansion.		

ROBOTS	List	SALE
ANDROBOT		
Slave to your computer		
TOPO III	1595.00	1485.00
Accessories		CALL
TOPO III moves and talks very well. It uses LOGO-like commands and a good text-to-speech system.		
RB ROBOT		
Download from your computer		
RB5X	2295.00	2058.00
Accessories		CALL
Program it in Tiny BASIC or use a SAVVY to Tiny BASIC "compiler". RB5X has bumpers and sonar, voice with sound effects, arm, easy-to-use programming language, additional senses are available. It is good quality. Our two demos have seen hard use and haven't broken down yet!		
ARCTEC		
GEMINI	6995.00	6506.00
A marvel. Three CPUs, voice recognition, goal-oriented navigation, on-board disk drive, detachable keyboard, and more.		

TECHNICAL INFO—WE CAN HELP. CALL

Shipping Over $200 Add 4%. $200 And Under Add 5%
Cash With Order Deduct 3%. N.M. Orders Add 5% Sales Tax
Allow 4 Weeks For Delivery.

RIO GRANDE ROBOTICS

A Division of Mobile Intelligence Corporation
1595 W. Picacho #28, Las Cruces, N.M. 88005. Tel. (505) 524-9480

Ode to the Typographical Error

The typographical error is a slippery thing and sly,

You can hunt till you are dizzy, but it somehow will get by.

'Til the forms are off the presses, it is strange how still it keeps,

It shrinks down in a corner and it never stirs or peeps.

That typographical error, too small for human eyes,

'Til the ink is on the papre, when it grows to mountain size.

The boss, he stares with horror, then he grabs his hair and groans.

The copyreader drops her head upon her hands and moans.

The remainder of the issue may be clean as clean can be...

But the typographical error is the only thing you see.

Back Issues Available:

Ordering Information: Back issues of The Computer Journal are $3.25 in the U.S. Canada, and $5.50 in other countries (air mail postage included). Send your complete name and address with payment to The Computer Journal, PO Box 1697, Kalispell, MT 59903. Please allow three to four weeks for delivery.

Volume 1, Number 1 (Issue #1):
- *The RS-232-C Serial Interface, Part One*
- *Telecomputing with the Apple][: Transferring Binary Files*
- *Beginner's Column, Part One: Getting Started*
- *Build an "Epram"*

Volume 1, Number 4 (Issue #4):
- *Optoelectronics, Part One: Detecting, Generating, and Using Light in Electronics*
- *Multi-user: An Introduction*
- *Making the CP/M User Function More Useful*
- *Build a Hardware Print Spooler, Part Three: Enhancements*
- *Beginner's Column, Part Three: Power Supply Design*

Volume 2, Number 1 (Issue #5):
- *Optoelectronics, Part Two: Practical Applications*
- *Multi-user: Multi-Processor Systems*
- *True RMS Measurements*
- *Gemini-10X: Modifications to Allow both Serial and Parallel Operation*

Volume 2, Number 2 (Issue #6):
- *Build a High Resolution S-100 Graphics Board, Part One: Video Displays*
- *System Integration, Part One: Selecting System Components*
- *Optoelectronics, Part Three: Fiber Optics*
- *Controlling DC Motors*
- *Multi-User: Local Area Networks*
- *DC Motor Applications*

Volume 2, Number 3 (Issue #7):
- *Heuristic Search in Hi-Q*
- *Build a High-Resolution S-100 Graphics Board, Part Two: Theory of Operation*
- *Multi-user: Etherseries*
- *System Integration, Part Two: Disk Controllers and CP/M 2.2 System Generation*

Volume 2, Number 4 (Issue #8):
- *Build a VIC-20 EPROM Programmer*
- *Multi-user: CP/Net*
- *Build a High-Resolution S-100 Graphics Board, Part Three: Construction*
- *System Integration, Part Three: CP/M 3.0*
- *Linear Optimization with Micros*
- *LSTTL Reference Chart*

Volume 2, Number 5 (Issue #9):
- *Threaded Interpretive Language, Part One: Introduction and Elementary Routines*
- *Interfacing Tips and Troubles: DC to DC Converters*
- *Multi-user: C-NET*
- *Reading PCDOS Diskettes with the Morrow Micro Decision*
- *LSTTL Reference Chart*
- *DOS Wars*
- *Build a Code Photoreader*

Volume 2, Number 6 (Issue #10):
- *The FORTH Language: A Learner's Perspective*
- *An Affordable Graphics Tablet for the Apple][*
- *Interfacing Tips and Troubles: Noise Problems, Part One*
- *LSTTL Reference Chart*
- *Multi-user: Some Generic Components and Techniques*
- *Write Your Own Threaded Language, Part Two: Input-Output Routines and Dictionary Management*
- *Make a Simple TTL Logic Tester*

Volume 2, Number 7 (Issue #11):
- *Putting the CP/M IOBYTE To Work*
- *Write Your Own Threaded Language, Part Three: Secondary Words*
- *Interfacing Tips and Troubles: Noise Problems, Part Two*
- *Build a 68008 CPU Board For the S-100 Bus*
- *Writing and Evaluating Documentation*
- *Electronic Dial Indicator: A Reader Design Project*

Volume 2, Number 8 (Issue #12):
- *Tricks of the Trade: Installing New I/O Drivers in a BIOS*
- *Write Your Own Threaded Language, Part Four: Conclusion*
- *Interfacing Tips and Troubles: Noise Problems, Part Three*
- *Multi-user: Cables and Topology*
- *LSTTL Reference Chart*

Volume 2, Number 9 (Issue #13):
- *Controlling the Apple Disk][Stepper Motor*
- *Interfacing Tips and Troubles: Interfacing the Sinclair Computers, Part One*
- *RPM vs ZCPR: A Comparison of Two CP/M Enhancements*
- *AC Circuit Anaysis on a Micro*
- *BASE: Part One in a Series on How to Design and Write Your Own Database*
- *Understanding System Design: CPU, Memory, and I/O*

Issue Number 14:
- *Hardware Tricks*
- *Controlling the Hayes Micromodem II From Assembly Language*
- *S-100 8 to 16 Bit RAM Conversion*
- *Time-Frequency Domain Analysis*
- *BASE: Part Two*
- *Interfacing Tips and Troubles: Interfacing the Sinclair Computers, Part Two*

Issue Number 15:
- *Interfacing the 6522 to the Apple][and][e*
- *Interfacing Tips and Troubles: Building a Poor-Man's Logic Analyzer*
- *Controlling the Hayes Micromodem II From Assembly Language, Part Two*
- *The State of the Industry*
- *Lowering Power Consumption in 8" Floppy Disk Drives*
- *BASE: Part Three*

Issue Number 16:
- *Debugging 8087 Code*
- *Using the Apple Game Port*
- *BASE: Part Four*
- *Using the S-100 Bus and the 68008 CPU*
- *Interfacing Tips and Troubles: Build a "Jellybean" Logic-to-RS232 Converter*

Issue Number 17:
- *Poor Man's Distributed Processing*
- *Base: Part Five*
- *FAX-64:Facsimile Pictures on a Micro*
- *The Computer Corner*
- *Interfacing Tips and Troubles: Memory Mapped I/O on the ZX81*